1. Introduction ..... Page 3
1.1. Les sites de presses actuels ..... Page 3
1.2. Une présentation plus accessible ..... Page 4

2. Présentation de notre site ..... Page 4
2.1. Les normes à respecter ..... Page 4
2.2. Un site réutilisable / facile à modifier ..... Page 5
2.3. Description du modèle ..... Page 5
2.3.1. Les acteurs ..... Page 5
2.3.2. Les informations échangées ..... Page 5
2.4. Présentation globale / navigation type sur le site ..... Page 7

3. Le crawling ..... Page 11
3.1. Choix technologiques ..... Page 11
3.2. Récupérer le graphe de liens d'un site ..... Page 12
3.2.1. Etude des sources des sites de presse (Html/PHP) ..... Page 12
3.2.2. Choix conceptuels ..... Page 14
3.3. Différencier articles / rubriques ..... Page 16
3.3.1. Phase d'études et de tests ..... Page 16
3.3.2. Choix conceptuels ..... Page 17
3.4. La Base de données, centre de l'interaction entre équipe ..... Page 19
3.4.1. Présentation, choix ..... Page 19
3.4.2. Diagramme UML ..... Page 20

4. L'indexation ..... Page 20
4.1. Stratégie d'indexation d'un site de presse ..... Page 20
4.1.1. Indexer le titre d'un article ..... Page 21
4.1.2. Indexer le texte d'un article ..... Page 23
4.2. Stratégie d'indexation des termes ..... Page 23
4.2.1. Les motifs de mots utilisés ..... Page 23
4.2.2. Traitements des mots / insertion dans la BD ..... Page 25

5. La conception de l'interface, choix ..... Page 27
5.1. Création d'une architecture ..... Page 27
5.2. Requête sur les termes, et articles ..... Page 35
5.3. Vers un site dynamique, insertion des requêtes ..... Page 37
5.4. Finalisation ..... Page 37

6. Les outils utilisés pour l'interface ..... Page 38
6.1. Le dynamisme du site avec PHP / MySQL ..... Page 38
6.2. A la découverte de SVG ..... Page 38
6.2.1. SVG basé sur XML ..... Page 38

6.2.2. L'animation et le scripting ..... Page 41

7. Difficultés rencontrées ..... Page 42

8. Conclusion / travail en groupe ..... Page 43
8.1. Le travail en groupe ..... Page 43
8.2. Notre point de vue sur le projet ..... Page 43
8.3. Améliorations possibles ..... Page 44

9. Référence ..... Page 45

10. Annexes, manuel utilisateur ..... Page 46

1. Introduction

Le but de notre travail, qui s'effectue à deux binômes, est de faciliter le contenu des sites de presse. Les « Unes » des sites actuels sont très chargées et tiennent sur plusieurs écrans. Beaucoup d'informations sont amenées au lecteur et il n'est pas forcément à même de se repérer dans cette amas d'informations qui rend le site peu lisible. Trop de directions possibles (liens hypertextes) sont proposées à l'internaute qui se retrouve ainsi « noyé » dans le contenu du site.

On a donc remarqué un manque d'accessibilité à l'information qui nous a amené à travailler sur un nouveau concept de présentation de cette information.

Dans la suite de ce rapport nous allons vous expliquer comment nous avons pu créer une nouvelle présentation en utilisant la navigation dans les graphes de termes et dans les graphes des articles ainsi que par une étude préalable des sites de presse donnant lieu au crawling et à l'indexation des termes et articles.

1.1. Les sites de presse actuels

Les sites presse actuels adoptent tous ou presque le même type d'interface graphique, qui est calquée sur la version papier du journal. On se retrouve donc avec des pages au contenu abondant et non clair pour le lecteur. On voit tout de suite que l'information ne tient pas sur une page, que l'utilisation de la « scroll bar » est indispensable pour voir la totalité de la page. On présente au lecteur un nombre trop important de liens hypertextes qui ne lui permet pas de se retrouver correctement. On peut retrouver ce type d'interface sur tous les grands sites de presse tels que « Le Monde », « Le Figaro »…

Fig.1. Exemple de « Une » du site « Le Monde »

1.2. Une présentation plus accessible

Comme on vient de le décrire ci-dessus, trop d'informations est présenté au lecteur. Pour tenter de présenter cette information, nous avons décidé de faire naviguer le lecteur dans les graphes de termes et graphes d'articles.

Les termes sont des mots ou des groupes de mots qui se répètent dans un article. Bien entendu, il ne faut pas s'occuper des articles, des déterminants par exemple. Donc dans chaque article on peut extraire les termes les plus récurrents et ainsi les stocker pour avoir facilement accès à l'information car ils seront les mots clefs de l'article. On peut relier donc les termes aux articles.

Dans plusieurs articles, on va retrouver des termes identiques et on pourra alors relier ces articles par ces termes communs.

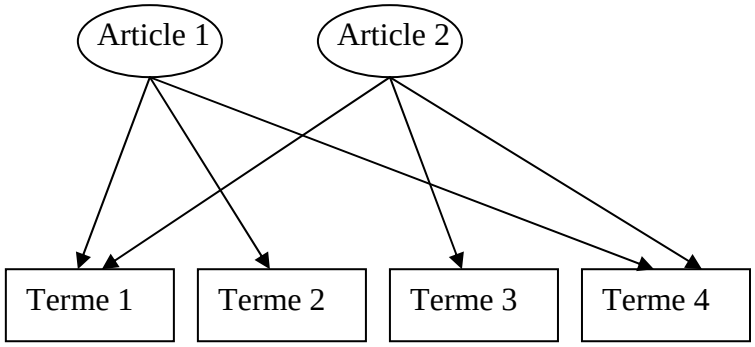


Fig.2. Le terme 1 et le terme 4 relie les deux articles

2. Présentation de notre site

2.1. Les normes à respecter

Bien évidemment, le but de notre travail est de refaire la présentation d'une autre manière, il ne faut pas reprendre ce qui existe déjà.

Comme on vient de le voir, nous essayons de travailler sur les termes, les articles et leurs liens donc nous allons nous attacher à faire une présentation de l'information autour de ces liaisons entre termes et articles.

Pour représenter ces liaisons nous devons utiliser des dessins dynamiques.

Nous nous sommes accordé pour garder l'accès à l'information par le biais des rubriques.

Notre site doit pouvoir accueillir n'importe quel site de presse dans n'importe quelle langue.

Le cahier des charges est assez libre car c'est notre imagination qui doit nous amener à une interface agréable et efficace.

2.2. Un site réutilisable / facile à modifier

Le site a été conçu pour qu'il soit réutilisable sur un grand nombre de sites de presse.

La source des données pour l'interface graphique est une base de données dont le schéma a été défini par l'équipe entière. Nous pouvons crawler la plupart des sites de presses actuels.

L'ensemble des fonctions pour dessiner les courbes entre termes, dessiner des graphiques, déplacer des éléments SVG (translation par exemple)… sont réutilisables facilement. Elles se situent dans un fichier annexe qui constitue une librairie.

L'ensemble de nos choix sur les couleurs peut être modifié par le biais d'un fichier externe. On peut alors modifier la couleur du texte, du fond d'écran…

2.3. Description du modèle

Dans cette partie, nous allons vous présenter les acteurs concernés par notre site, ainsi que les informations échangées entre ces acteurs et le système.

2.3.1 Les acteurs

L'utilisateur est un acteur important. Ses relations avec le site sont du type « questions / réponses ». Il interroge le système par l'intermédiaire du moteur de recherche et obtient des résultats qui sont en rapport avec sa requête.

Une autre catégorie d'acteur est également concernée par notre site. Il s'agit de l'administrateur. Il intervient sur le système mais celui-ci ne lui répond pas.

2.3.2 Les informations échangées

Les informations échangées entre le système et les acteurs correspondent en fait à des réceptions et à l'émission de messages, entre les acteurs et le système.

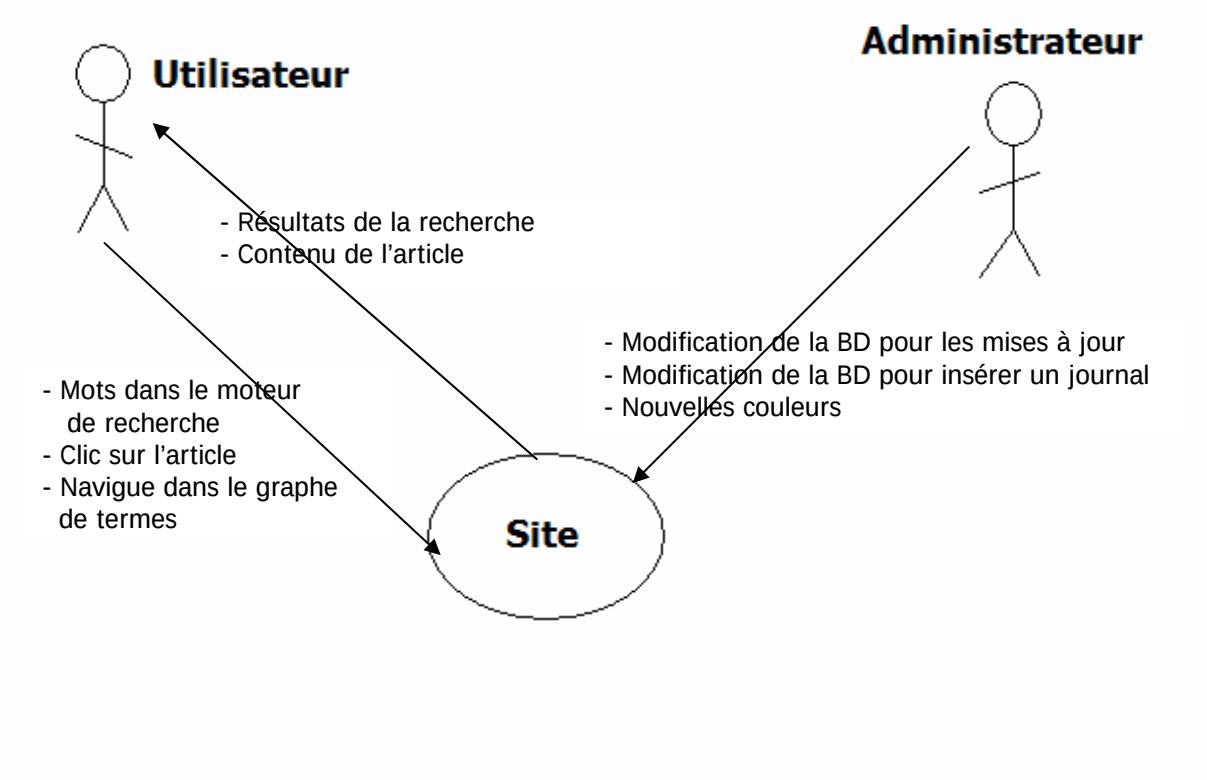
L'utilisateur se sert du site pour effectuer des recherches. Il émet un message qui correspond aux mots qu'il a entré dans le moteur de recherche. Le système lui renvoie alors les articles qui sont en relation avec ces mots. Il peut également recevoir d'autres types d'information comme l'affichage des liens entre les termes et les articles, et celui du contenu des articles.

Les messages entre le système et l'administrateur ne vont que dans un sens : ils sont envoyés par l'administrateur et reçus par le système. Ces messages peuvent avoir un rapport avec la base de données ou l'interface. Pour ce qui concerne la base de donnée, l'administrateur a la possibilité de mettre à jour certaines données (notamment les informations importantes qui arrivent en cours de journées). Il peut également introduire de nouveaux journaux. Pour l'interface, il peut en modifier les couleurs.

Tableau récapitulatif

Cas d'utilisation	Acteurs	Informations échangées
Recherche	Utilisateur	**Emet :** recherche en tapant des mots dans le moteur de recherche **Reçoit :** résultats de la recherche
Lecture d'un article	Utilisateur	**Emet :** clic sur un article **Reçoit :** contenu de l'article
Mise à jour	Administrateur	**Emet :** modifications de la base de données
Insertion de nouveaux journaux	Administrateur	**Emet :** modifications de la base de données
Modification des couleurs de l'interface	Administrateur	**Emet :** nouvelles couleurs

Diagramme



2.4. Navigation type sur notre site

✓ Arrivée sur le site

Avant de pouvoir effectuer des recherches à l'aide de notre site, l'utilisateur arrive sur une page où il doit choisir le journal qu'il veut consulter. Il peut également, sur cette page, obtenir des informations sur les créateurs du site.

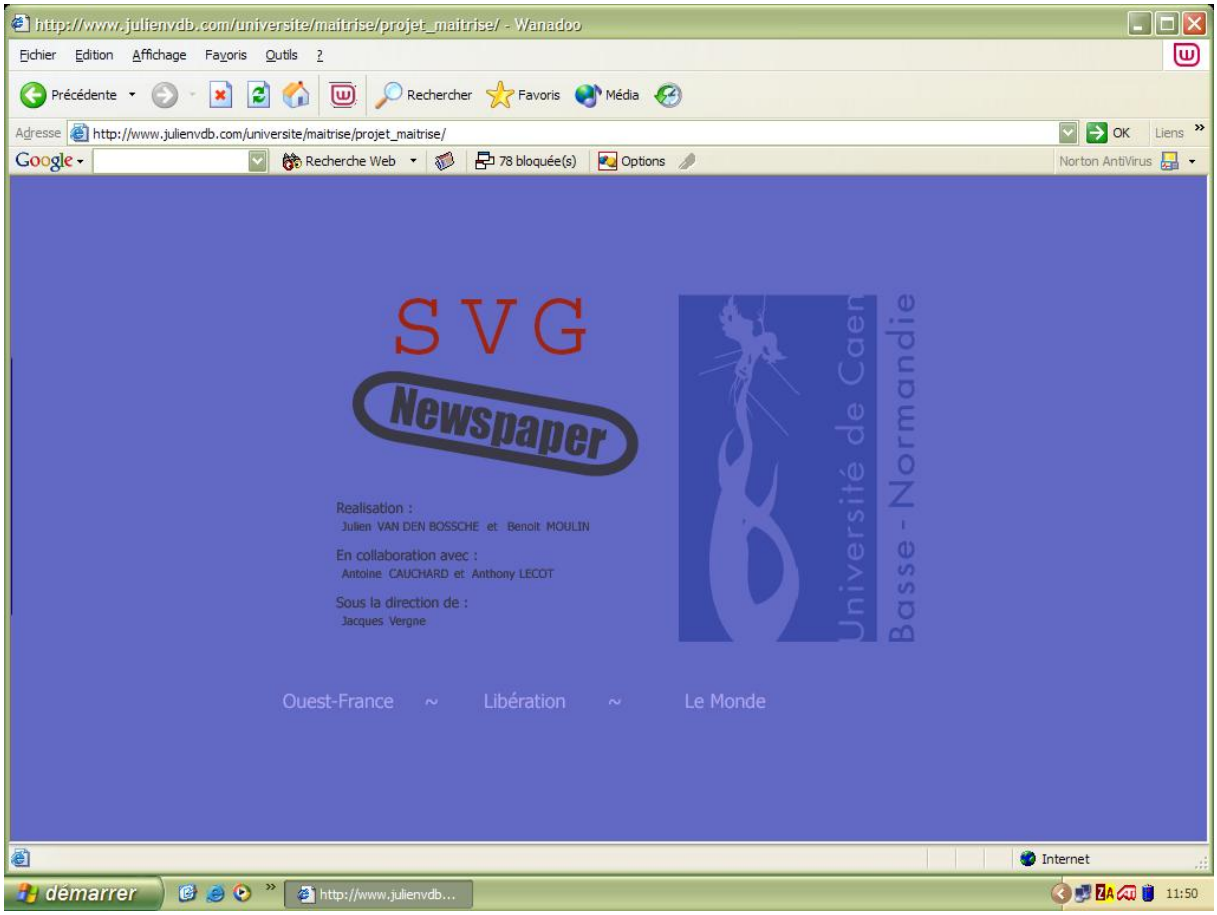


Fig.3. Choix du journal

✓ Page de Une

Une fois le journal choisi, l'utilisateur arrive sur la page de Une avec les termes les plus fréquents qui apparaissent aujourd'hui.

Fig.4. La page de Une

✓ Recherche classique

Une fois le journal choisi, l'utilisateur entre les termes de sa recherche dans le champ prévu à cet effet. Ensuite, il clique sur « OK » et voit les résultats de la recherche apparaître : d'une part, les titres des articles qui sont en relation avec les mots tapés par l'utilisateur, d'autre part, les principaux termes contenus dans ces articles. En passant le curseur de la souris sur les articles et sur les termes, l'utilisateur peut alors observer dans quels articles il pourra trouver les différents termes. Enfin, il lui suffit de cliquer sur un article pour pouvoir le lire.

Si l'utilisateur est intéressé par d'autres termes qui résultent de sa recherche, il peut cliquer dessus pour lancer de nouvelles recherches à partir de ces termes.

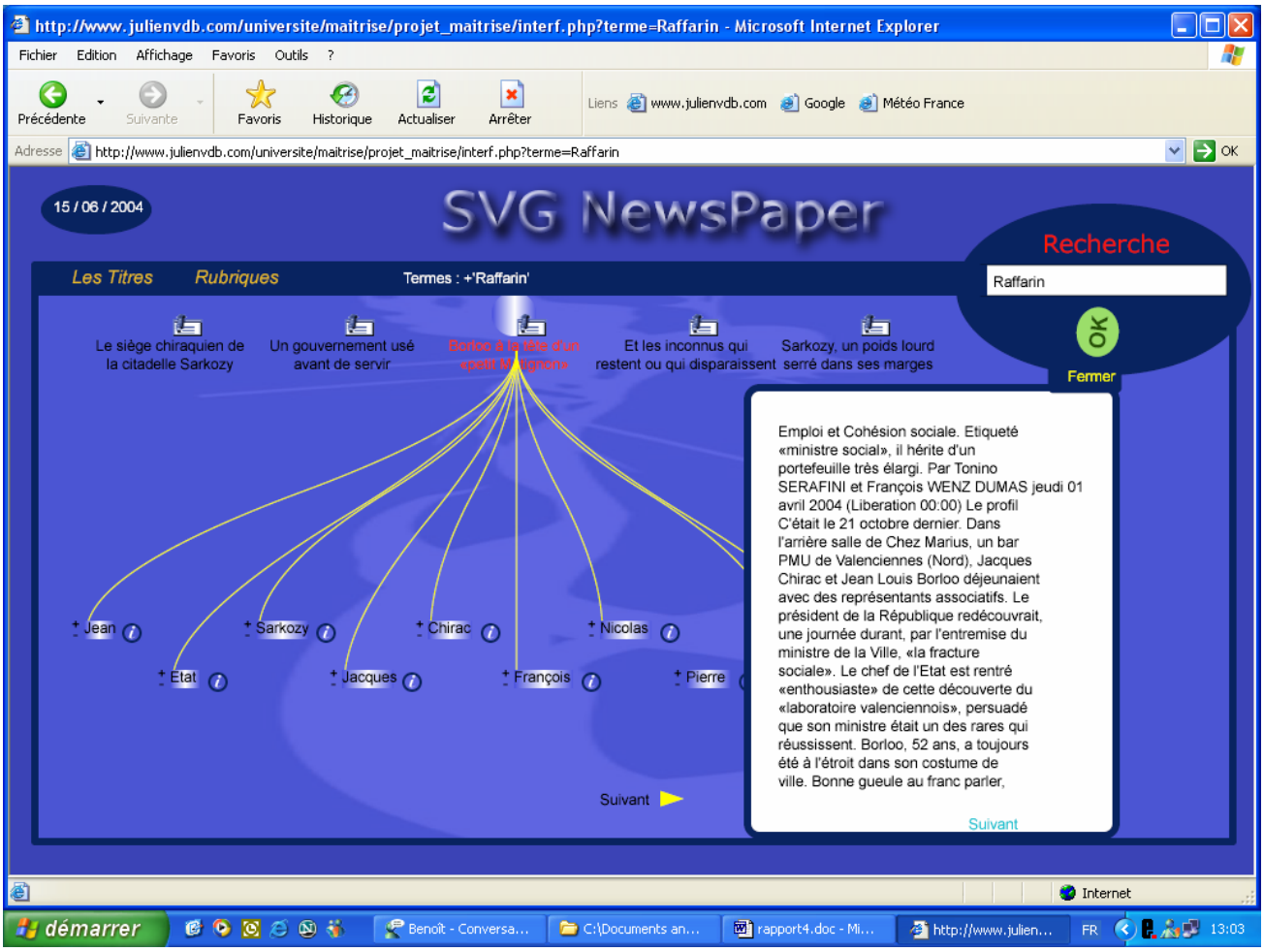


Fig.5. Ecran montrant les résultats d'une recherche ainsi qu'un article

✓ <u>Recherche approfondie</u>

L'utilisateur peut affiner sa recherche. En effet, si certains termes obtenus ne l'intéressent pas, il peut les retirer par l'icône « - » qui se trouve à droite du terme (pas très visible sur la copie d'écran). Dans ce cas, une nouvelle recherche est lancée avec les mêmes mots d'origine (tapés par l'utilisateur), avec les mêmes articles, mais sans les termes inintéressants pour l'utilisateur. Il peut aussi ajouter un des termes à sa recherche en cliquant sur l'icône « + » à droite du terme. Quand le curseur de la souris passe sur l'un des icônes une info bulle lui dit l'action envisageable (par ex : enlever ce terme de la recherche).

L'utilisateur peut également obtenir des informations sur la fréquence des termes dans les articles. En passant le curseur de la souris sur l'icône situé à côté de chaque terme, un « camembert » apparaît avec des pourcentages qui correspondent au nombre d'occurrences du terme dans les articles qui le contiennent.

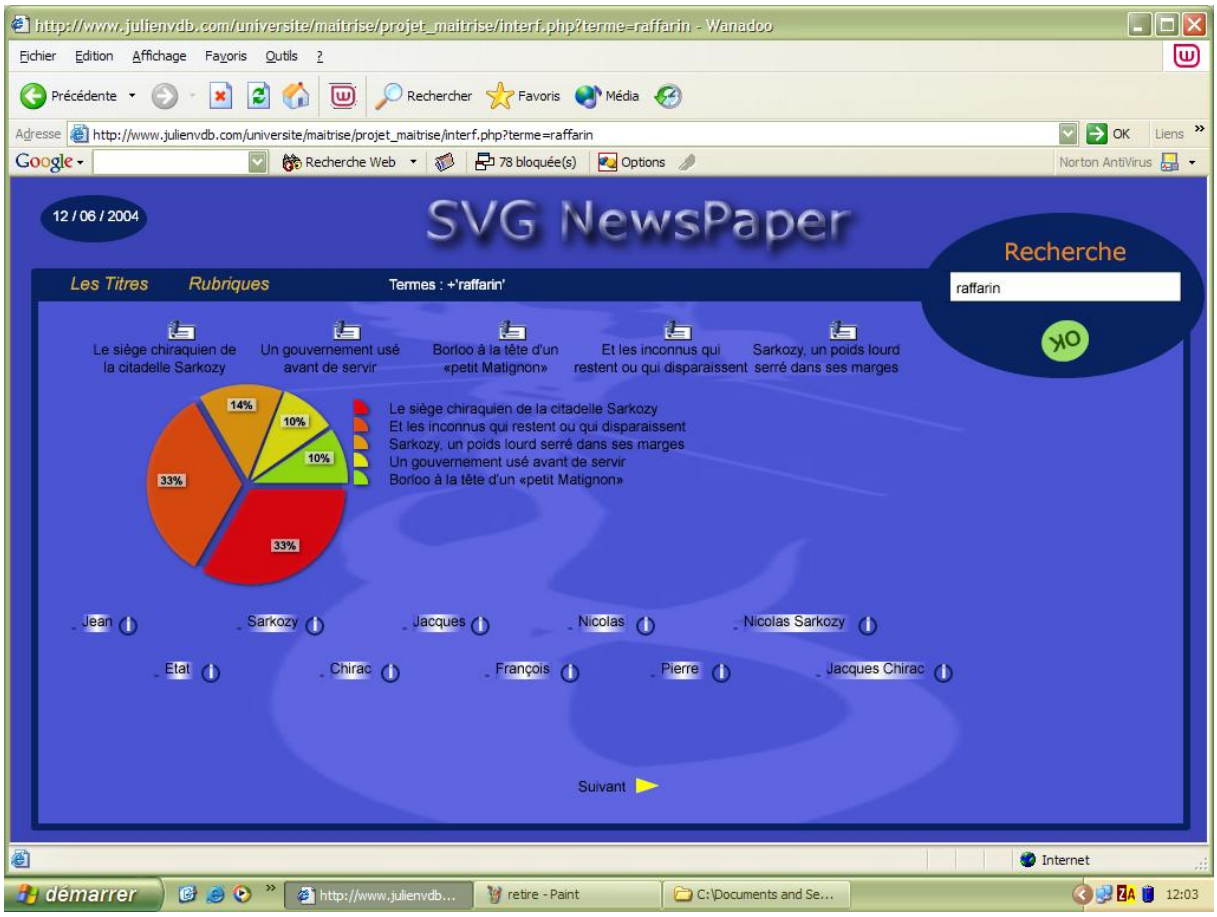


Fig.6. Présence du mot « Etat » dans les différents articles

✓ <u>Rubrique</u>

L'utilisateur a la possibilité de consulter le site par le biais des rubriques. Cette consultation n'est pas très pertinente car le crawling des rubriques est difficile. Sur certains sites on obtient des résultats satisfaisants.

Fig.7. Consultation par rubrique

Vous avez à présent une vision globale de notre site, nous allons donc pouvoir vous expliquer comment nous avons réaliser ce travail en vous exposant les différentes étapes de la conception.

3. Le crawling

Cette partie du projet consiste à récupérer le graphe de liens d'un site, c'est à dire récupérer l'arborescence interne du site. Au cours de ce crawling, nous allons également stocker des variables utiles afin de différencier les articles des rubriques.

3.1 Choix technologiques

Pour faire tourner nos algorithmes de crawling et d'indexation, nous utilisons PHP qui est un langage de script multi-plateformes. PHP possède beaucoup d'avantages par rapport à ces principaux concurrents que sont Java (JSP) et ASP, étant tous les trois les principaux langages dynamiques utilisés sur Internet. Pour commencer, PHP est gratuit et open-source, contrairement à ASP qui appartient à Microsoft et est lui payant. De plus, la syntaxe de PHP est proche de celle du C et c'est un langage de haut niveau, c'est à dire que beaucoup de fonctions existent déjà, sans avoir à les reprogrammer. Enfin, les avantages de PHP par rapport à JSP, c'est qu'il s'intègre très bien avec un Système de Gestion de Base de Données (SGBD) comme MySQL ou PostgreSQL, et que les temps d'exécutions de ses scripts sont

plus rapides que ceux des scripts JSP. Un autre avantage par rapport à Java est que PHP gère plus facilement les expressions régulières, principaux outils de notre programme. Cela s'explique par le fait que PHP a grandement remplacé le langage PERL qui était très pratique dans la manipulation de chaînes de caractères.

Le système de gestion de base de données que nous avons choisi est MySQL. Tout comme PHP, il est gratuit et en plus il s'intègre vraiment bien avec celui-ci. Outre Oracle qui est payant et surtout très cher, nous aurions pu utiliser PostgreSQL, un autre SGBD utilisant SQL comme langage. PostgreSQL est beaucoup plus évolué que MySQL, il est gratuit lui aussi. En effet, PostgreSQL est beaucoup plus orienté objet et dispose d'un langage de procédures qui lui est propre. Mais à la différence de MySQL, la communauté d'utilisateurs de PostgreSQL est encore très peu développée, mais surtout l'exécution de requêtes simples est beaucoup plus lente qu'avec MySQL. Or dans notre programme, nous n'utilisons que des requêtes simples.

3.2. Récupérer le graphe de liens d'un site

Afin de cerner les moyens que nous allions pouvoir mettre en œuvre pour accomplir cette tâche, nous avons étudié les sources de différents sites de presses. Nous avons ensuite essayer de comprendre l'arborescence globale d'un site : son graphe de liens. Une fois cette phase d'étude effectuée, nous avons fait des choix conceptuels qui nous ont permis d'obtenir la fonction requise.

3.2.1. Etude des sources des sites de presse/étude d'une page html

Depuis le début des années 1990, et l'explosion d'Internet, le langage HTML (Hyper Text Markup Language) est un standard incontournable et sa capacité à être étendue par d'autres langages de scripts permet désormais aux sites Internet d'être beaucoup plus dynamiques. Nous nous sommes donc intéressés dans un premier temps à l'impact que pouvez avoir ces différents langages sur la clarté et l'intégrité des sources HTML.

Les principaux langages de scripts utilisés par les sites de presses sont le PHP (Hypertext PreProcessor), l'ASP de Microsoft (Active Server Pages), le CGI (Common Gateway Interface), le JavaScript, issu du Java qui peut lui produire des applets (petit programmes java visible grâce à un navigateur), le Vbscript qui est lui issu du Visual Basic, et enfin le langage XML (eXtensible Markup Language) qui permet aux développeurs de créer leurs propres balises.

Pour récupérer le graphe de liens d'un site, il faut être capable de trier dans les sources du programme ce qui correspond à un lien. Il existe en fait deux méthodes pour représenter un lien :

```
<a paramètres href=url paramètres>texte de lien</a>
<javascript onClick=window.open(…)>
```

Cependant, la plupart des liens sont représentés avec la première méthode. Les paramètres présents à l'intérieur de la balise <a> peuvent être devant ou derrière l'url, ou les deux. Pour ce qui est de la deuxième représentation, des paramètres peuvent également être présents. De plus, après la chaîne **onClick**, on peut avoir n'importe quelle fonction java.

Après observation sur des sites de presse utilisant cette représentation, nous avons constaté que les développeurs l'utilisaient essentiellement pour des liens vers des images, des animations Flash… Nous n'aurons donc pas besoin de stocker ces liens. Il faudra donc trouver un moyen de récupérer tous les liens d'une page HTML à partir de la première représentation.

Un autre détail qui a son importance : toutes les URL (Universal Resource Locator) présents dans les balises <a> ne sont pas absolues. En effet, une URL correspond à une adresse sur le site ou à l'extérieur, et peut donc être relative ou absolue. Une adresse est relative si elle ne rappelle pas le chemin complet du fichier complet, mais seulement son adresse par rapport au chemin actuel. Il faudra donc rendre toutes les URL absolues pour éviter les redondances.

La dernière chose que nous avons observée concernant le source html, c'est la formulation des textes de liens. En effet, ils peuvent être sous plusieurs formes :

Du texte, plus ou moins long et bien formaté.

Aux Européennes, le PS confirme sa victoire des Régionales

Une image représentant le texte (notamment pour les rubriques).

Ou encore une image illustrant le texte de lien adjacent.

Étude de l'arborescence d'un site de presse

Un site web correspond à une arborescence finie, qui peut être plus ou moins profonde, mais qui restera toujours finie. On a constaté qu'à une profondeur trois, on touche quasiment tous les articles, et ce sur presque tous les sites. En effet, une profondeur correspond à un clic de l'internaute. En trois clics, l'internaute est entré dans une rubrique, puis une sous rubrique, et enfin vers un article. En effet, la plupart des sites de presses possèdent une arborescence de rubrique qui est égale à 2.

Par contre il ne faut pas oublier que sur un site web, il peut y avoir des liens externes, c'est-à-dire des liens qui pointent vers d'autres sites web. Il nous faudra donc trouver un moyen d'éviter de sortir de la structure interne du site, sans quoi la phase de crawling ne manquerait pas de parcourir le World Wide Web.

Dans le graphe de liens d'un site web, on trouve des fichiers possédant différentes extensions (html, php, asp, jpg, exe, zip, pdf, cgi, xml, xhtml…). Il nous faudra donc décider quels types de fichiers nous allons stocker dans notre graphe, et quels types nous allons exclure au cours d'une phase de crawling.

Un site de presse, au même titre que la plupart des sites webs commerciaux, a la particularité de rappeler certains liens importants sur presque toutes les pages (rubriques, pages d'accueil…). Cette fonctionnalité est d'ailleurs inévitable pour un gain de temps évident de navigation pour l'internaute. Il nous faudra donc également trouver une parade à cela, pour ne pas étudier une page déjà crawlée !

3.2.2. *Choix conceptuels*

Tout d'abord, il nous a fallut coder une fonction pour obtenir le tableau de liens d'une page HTML. Pour cela, nous avons utilisé la fonction php *preg_match_all()*. C'est en fait une fonction issue du langage PERL et qui utilise les expressions régulières pour trouver toutes les occurrences d'une chaîne de caractères dans une autre. Voici donc le motif d'expression régulière que nous avons utilisé pour obtenir le tableau de lien :

```
/(<[a,A] (.* ?href=|.* ?HREF=))([^>]*)(>)((.* ?[\n]*)*)(<\/[a,A]>)/
```

Ce motif prend en compte la casse de l'expression, les sauts de lignes éventuels, et les paramètres qui peuvent se trouver à l'intérieur de la balise ouvrante. Ensuite, nous savons que la troisième colonne du tableau retourné correspond à l'URL du fichier pointé par le lien. Nous savons également que la cinquième colonne correspond au texte de lien.

Ensuite, il nous faut rendre les URL absolues. Pour cela, nous avons développé un petit programme qui prend en compte la structure de l'URL trouvée, puis concatène le chemin du fichier où se trouvait le lien.

```
Si (pas « http » dans l'adresse)
   Si (« / » en premier caractère)
         ajoute l'url du site devant l'adresse
   Sinon
         reconstitution de l'url absolue
Sinon
   c'est bon, on nettoie juste l'adresse
```

Il nous faut également faire un travail d'affinage sur les textes de liens. Nous avons décidé d'exploiter uniquement le texte de lien sous format texte. Nous avons donc programmé un petit script qui nettoie le texte de lien pour ne garder que le texte intéressant. Si le texte de lien n'est qu'une image, nous essayons malgré tout de prendre la légende comme texte de lien (alt ou name). Nous conservons aussi l'original du texte de lien dans un champ de la table.

Une fois ces fonctions codées, il nous reste à coder la boucle principale du crawling, qui va stocker le graphe de liens dans la base de données MySQL. Cette boucle devra en fait commencer par stocker le tableau de liens de la une, puis automatiquement créer et stocker les tableaux de liens de chaque lien du premier tableau de liens. Et ainsi de suite jusqu'à ce qu'une condition soit vérifiée. Pour avoir une boucle qui s'arrête, il nous faut respecter les conditions citées dans la partie précédente.

Pour éviter de sortir du site, et de stocker des fichiers qui possèdent des extensions non voulues, nous avons défini une méthode StockOuPas, qui fait le tri. Pour les extensions, nous

avons simplement utilisé une expression régulière. Si l'extension est dans l'ensemble suivant, on n'enregistre pas le lien dans la base de données.

```
{mailto, javascript, pdf, jpg, bmp, gif, jspx, swf}
```

Cette liste est exhaustive et permet un bon tri. Si l'on met toutes les extensions d'images possibles, la phase de test prendra plus de temps. Il faut toujours garder à l'esprit de rendre le code le plus optimal possible.

Pour ce qui est des liens externes, nous avons ajouté dans la base de données une table sites. Cette table contient pour chaque site : son nom, son url, et une chaîne de caractères persistants. Cette chaîne correspond à l'url de la page web privée des caractères du départ.

Exemple pour le site de Ouest-France :

```
URL : http://www.ouest-france.fr
Chaîne persistante : ouest[-]*france.fr
```

Le cas de Ouest-France est particulier. En effet, la chaîne persistante est un motif d'expression régulière. Il s'est avéré que Ouest-France utilisait la notation avec et sans tiret ! Dans tous les autre cas, c'est juste l'URL privée de «http://www.» Pour naviguer dans un site, l'URL peut-être constitué ainsi :

```
http://tv.ouestfrance/fichier.asp
http://www.ouest-france/tv/fichier.asp
```

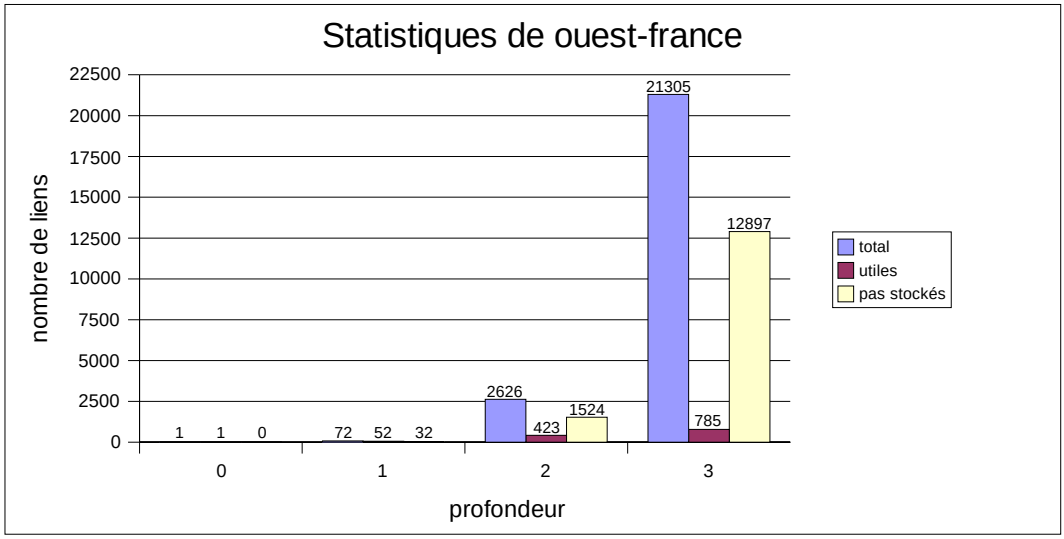
Pour ne pas crawler des fichiers déjà enregistrés dans la base de données, on a ajouté un champ **clone** dans la table **liens**. Quand le crawler enregistre le tableau de liens dans la base, il vérifie pour chaque lien que son URL n'est pas présent dans la base. Si c'est le cas, il met le champ clone à '1', sinon, il le met à '0'. Ainsi, le crawler ne crawle que les liens qui ont le champ clone à '1'. Le crawler s'arrêtera donc quand, pour une profondeur donnée, il n'y a que des clones !

Pour la version finale du projet, nous nous sommes limités à une profondeur 3 de crawl. En effet, une bonne partie des articles est crawlée à la profondeur 3. Cette profondeur est cependant paramétrable, pour permettre de personnaliser le crawl en fonction du site.

Voici donc le schéma de la table liens, que nous remplissons au cours de cette première phase. Nous ajoutons bien entendu le niveau du lien et l'identifiant de son père pour pouvoir le situer. Le reste nous l'avons expliqué précédemment.

liens
+id_lien: int(6)
+nivo: tinyint(1)
+pere: int(6)
+url: varchar(200)
+texteLien: text
+original: text
+clone: tinyint(1)

Après crawling sur le site de Ouest-France, voilà l'arborescence, avec le nombre de liens total, le nombre de liens utiles, et le nombre de liens externes ou vers des pages d'un format proscrit.



Le temps d'exécution d'un tel crawl est de 2171 secondes, soit à peu près 35 minutes.

3.3. Différencier article / rubrique

Pour effectuer cette tâche, nous avons du dans un premier temps assimiler les différentes structures des sites de presses. Nous avons pour cela essayer différents tests sur des paramètres intéressants du graphe de liens.

3.3.1. Phase d'étude et de tests

Le premier élément de test que nous avons utilisé pour différencier les articles et les rubriques, c'est le nombre de liens sortants d'une page.

Tout d'abord, nous avons essayé de déterminer le nombre minimum de liens sortants pour une page, c'est-à-dire le nombre de liens qui sont présents sur toutes les pages. Ce nombre de liens correspond à la structure de navigation d'un site : le titre du journal tout en haut, la liste des rubriques sur le côté gauche, au milieu des articles brûlants, et à droite bien

souvent une liste plus longue d'articles de dernières minutes. Il s'est avéré que ce nombre est très difficile à chiffrer. En effet, il y a toujours des pages qui ne rappellent pas les rubriques. Pour une rubrique, le nombre de liens sortants sera forcément plus grand que pour un article. En effet, une rubrique possédera des liens pour toutes ses sous rubriques plus des liens pour des articles correspondants. Tandis qu'un article ne possédera bien souvent qu'un lien de retour vers sa rubrique, et quelques liens vers des articles ayant trait au même sujet.

Un autre facteur qui peut déterminer la différence entre un article et une rubrique, c'est le nombre d'occurrences d'une page dans le graphe de liens. On voit bien que le graphe de liens est là encore déterminant. Ce nombre d'occurrences est en fait le nombre de liens entrants pour cette page. Il est facile de remarquer qu'une rubrique sera appelée beaucoup plus de fois qu'un article. Une rubrique est en effet appelée par ses articles pour créer un lien de retour, ainsi que par ses sous rubriques pour la même raison. Tandis qu'un article n'est appelé que par sa rubrique, sur certains sites par des articles ayant trait au même sujet, et également par la une ou tête de rubrique s'il représente une information importante, ou récente.

Nous avons également pu remarquer un genre de rubriques qui ne nous intéresse pas pour le sujet de notre projet : les annuaires webs, ou encore les cotations boursières ! Il nous faudra donc éliminer les sous rubriques récurrentes de ces rubriques pour clarifier notre présentation de contenu du site.

3.3.2. Choix conceptuels

Trouver le minimum de liens sortants pour un site s'est avéré très dur à réaliser, nous avons donc trouvé une autre méthode : faire la moyenne des liens sortants, pour ensuite avoir une donnée de différenciation. Nous avons donc pris tous les liens qui n'étaient pas des clones, et les avons enregistrer dans une table **page**. Nous avons ajouté dans cette table la donnée *nbLiensSortants*, qui correspond exactement au nombre de liens sortants de cette page. Cette opération était aisée car il a suffit de compter le nombre de liens (clone ou pas) qui avait pour père la page ajoutée, et voilà.

Pour calculer le nombre de liens entrants, il suffisait de compter le nombre de clone, plus 1. Ce facteur est aussi déterminant. En effet, une rubrique sera appelée beaucoup de fois par ses articles et sous rubriques. Tandis qu'un article ne sera appelé que par sa rubrique, les articles liés, et peut-être la une si c'est un article récent.

Nous avons également utilisé une valeur importante: le plus petit nombre de liens sortants d'une page qui est supérieur à 10. Ce nombre nous permet de ne pas sélectionner des pages dont l'accès serait payant, ou même une pop up avec une image.

La méthode de différenciation s'effectue en deux phases:

- une phase qui transpose les liens non clonés de la table liens dans une table pages, avec ajout du nombre de liens sortants, du nombre d'occurrences de la page, et du niveau minimum où la page apparaît.
- une phase de réelle différenciation, en utilisant les données du site (moyenne de liens sortants, nombre d'occurrences, et minimum de liens sortants)

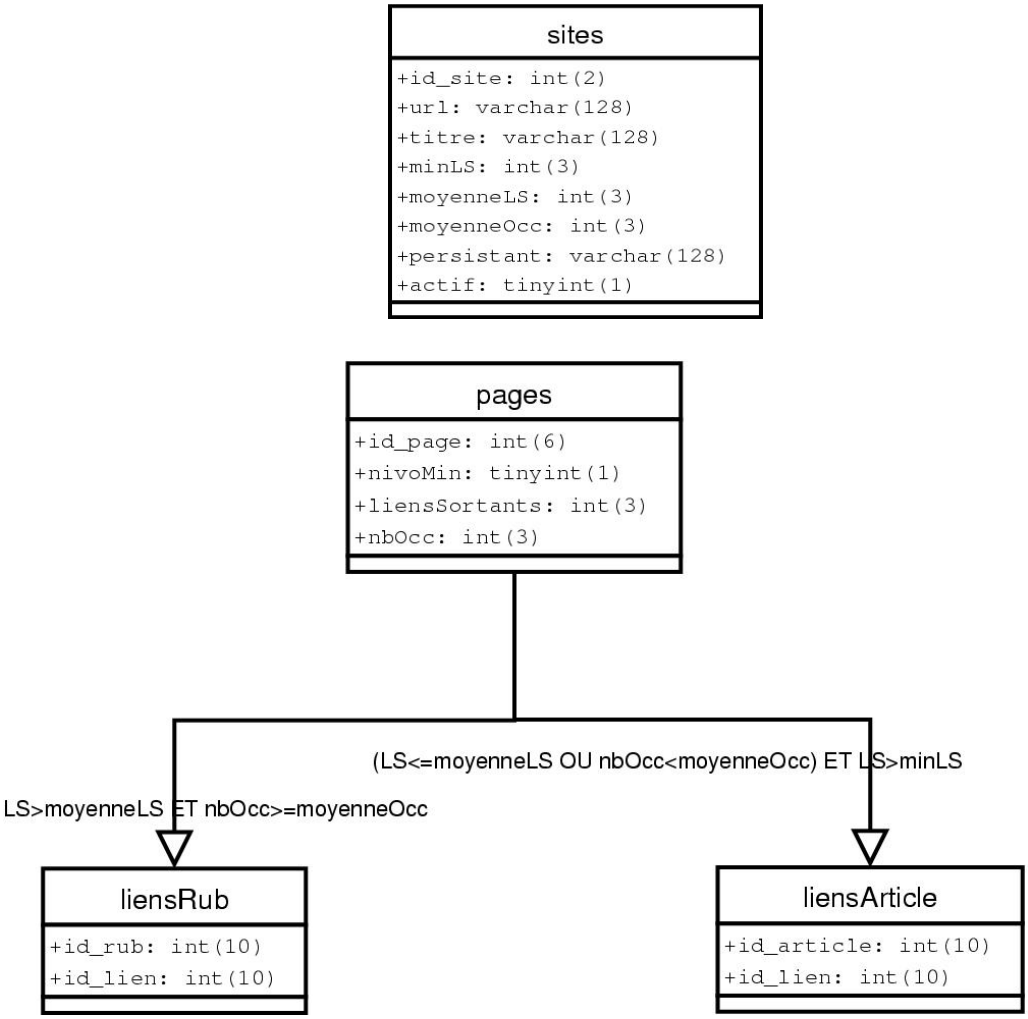
Pour la phase de différenciation, nous avons créer deux tables **liensArticle** et **liensRub**, qui contiendront uniquement les identifiants des pages associées. Il nous a donc suffit d'utiliser une fonction de sélection SQL avec les éléments de comparaisons suivants :

Article
```
- (minLiensSortants < liensSortants) OU (nbOcc < moyenneOcc)
- ET liensSortants > minLiensSortants
```

Le premier critère utilise la moyenne de liens sortants et la moyenne d'occurrences. Nous avons également ajouté dans le premier critère une clause qui enlève les pages dont le nombre de liens sortants est inférieur à un nombre minimum de liens sortants. En effet, si la page n'a pas assez de liens sortants, cela correspond bien souvent à une page payante, ou un pop-up, ce qui n'est donc pas intéressant pour nous.

Rubrique
```
- liensSortants > moyenneLiensSortants
- ET nbOcc >= moyenneOcc
```

Pour la sélection des rubriques, c'est exactement la clause inverse des articles, ce qui est tout à fait logique !
Voici donc le schéma des tables qui nous servent au cours de la phase de différenciation.



Le temps d'exécution de cette phase pour Ouest-France est de moins de cinq minutes, et renvoie 1162 pages intéressantes, composées de 1078 articles, et 38 rubriques. 46 liens ont étaient écartés du fait de leurs nombres de liens sortants insuffisants.

3.4. La Base de données, centre de l'interaction entre équipe

3.4.1. Présentation, choix

Après ces deux premières parties, il est devenu indispensable de présenter nos choix quant au schéma de la base de données. Ces choix furent motivés essentiellement par l'équipe de crawling/indexation, et ont étaient validés par l'équipe d'interfaçage. La base de données est réellement le centre du projet. En effet, elle est remplie par le crawling et l'indexation, et est utilisée par l'interface SVG.

On a déjà vu en détail le remplissage des tables **sites**, **liens**, **pages**, **liensRub**, et **liensArticle** dans la section précédente, ainsi que leurs attributs. Ces tables sont donc remplies tour à tour par le crawler. Seul la table **sites** est remplie par l'administrateur, pour y ajouter des sites Internets. Qui seront d'ailleurs modifiés par la suite pour y ajouter des statistiques.

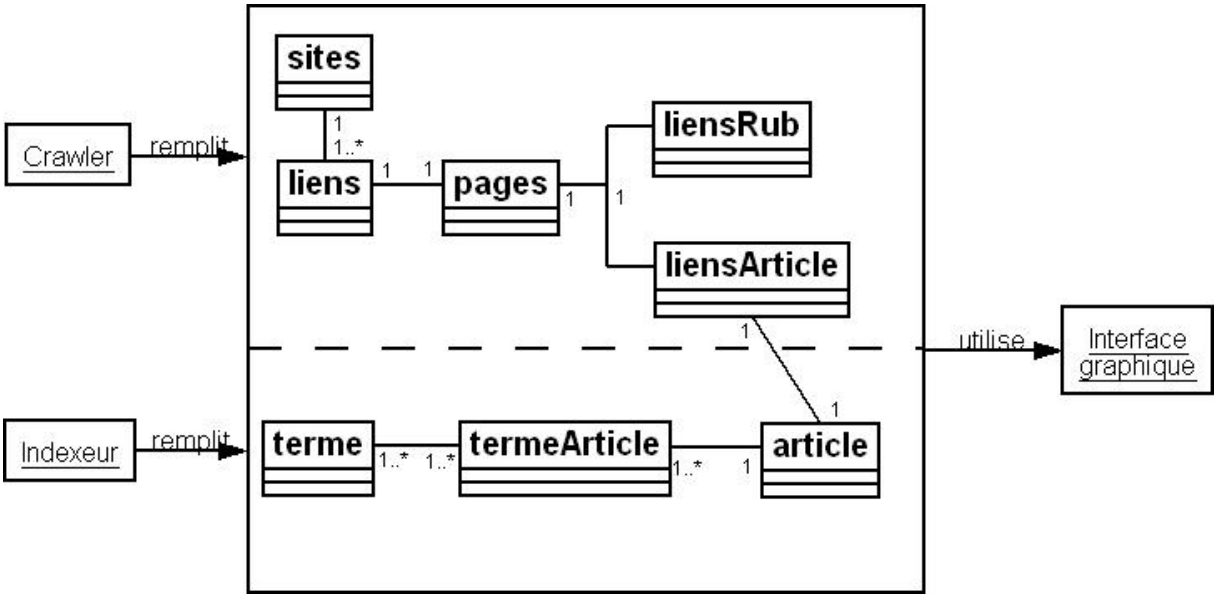
Les tables **liensRub** et **liensArticles** sont en fait des tables de relations. En effet, elles ne contiennent que les identifiants respectifs des liens et des rubriques ou articles. Elles nous servent pour différencier les articles des rubriques.

Pour la partie indexation, les tables **article**, **terme**, et **termeArticle** entrent en jeu. Nous en verrons les attributs à la fin de la partie suivante. La table **article** contient le titre de l'article, ainsi que son identifiant, et l'identifiant de la rubrique à laquelle chaque article appartient. Les termes de l'article sont stockés dans la table **terme**.

Là encore, nous avons une table de relation entre termes et articles : **termeArticle**.

Grâce aux remplissages de ces tables, l'interface de notre programme peut piocher dans la base de données afin de présenter le contenu du site. Et ainsi proposer à l'internaute une magnifique interface SVG, réalisée par le binôme d'interface.

3.4.2. Diagramme UML



4. Indexation d'un site de presse

4.1. La stratégie à adopter pour indexer un article de presse

4.1.1. Indexer le titre d'un article

Chaque article d'un site de presse possède un titre. Dans le but de présenter les articles à l'utilisateur, pour atteindre l'article en question, il devra cliquer sur son titre. Nous devons faire en sorte de récupérer le bon titre associé au bon article. Nous avons plusieurs stratégies pour identifier un titre par rapport à son article qui lui est propre. La première piste que nous avions suivie est d'isoler le titre par rapport à la page html de l'article. En effet, dans la plupart des pages html d'un site, il y a un couple de balises (*<title>,</title>*) dont le contenu est le titre de la page, que l'on veut récupérer. Car, comme nous l'avons vu précédemment, l'article d'un site de presse est dans le contenu d'une page html. Malheureusement, le contenu se trouvant entre ces deux balises titres, n'est pas toujours le titre de l'article correspondant. Pour la plupart des sites de presse, il représente plutôt le nom du journal. C'est à dire que ce contenu est statique, il ne change pas au fur et à mesure des pages. Donc cette solution ne peut pas nous convenir pour repérer un titre d'article. La seconde piste sur laquelle nous nous sommes penchés est d'essayer de trouver un titre d'article dans les informations contenues dans le lien qui pointe l'article en question. Nous avons vu qu'un lien, identifié par la balise <a> en html, possède plusieurs paramètres dans le contenu de cette balise. Outre son url et son texte de lien, il y a un paramètre *alt*, qui affiche, dans le navigateur de l'utilisateur par l'intermédiaire d'une bande jaune se plaçant sous le pointeur de la souris quand celui-ci est positionné sur le lien, une information concernant le lien. Pour quelques sites de presse, on peut y retrouver le titre de l'article qui est pointé par ce lien, mais pour une grande partie y est affiché l'url du lien, ou même parfois rien du tout. Donc nous ne pouvons pas récupérer le titre de cette manière. Finalement, nous avons opté pour la stratégie suivante : tout titre d'un article est en fait le texte de lien du lien qui le pointe. Comme nous l'avons vu précédemment, le texte de lien est contenu entre les deux balises html de liens (*<a>*). Il est donc facile

de l'isoler. De plus cette stratégie fonctionne pour tous les sites de presse. La récupération du texte de lien d'une page html par les expressions régulières se fait lors de la phase de crawling. Mais dans l'indexation on associe le titre (le texte de lien) à l'article correspondant. Nous verrons plus loin comment est récupéré l'article. A partir de la table **liens** de la base de données, on récupère les textes de liens de chaque article (cette phase à été réalisée dans le crawling).

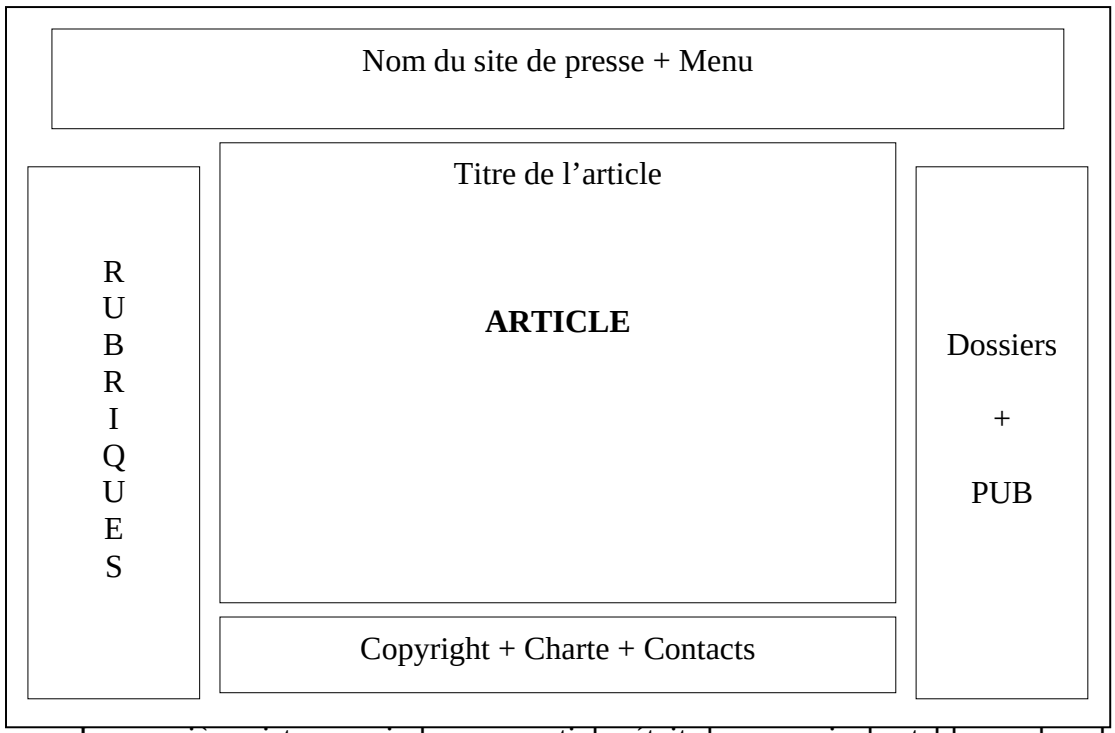
Non seulement le titre d'un article permet de résumer le contenu de l'article, ainsi que le ou les thèmes abordés, mais en plus il peut nous servir à filtrer encore plus les pages html. C'est à dire que, toutes les pages html d'un site de presse ne sont pas toutes des pages contenants un article. Les principales sélections ont eu lieu dans le crawling, nous proposons donc dans la phase d'indexation de ne lancer l'algorithme qui récupère les articles, que si le nombre de mots contenu dans le titre d'un article est supérieur ou égal à trois. Ce filtrage permet notamment de ne pas prendre en compte certaines pubs, les dossiers et les rubriques qui comportent une série de liens pointant vers des articles, mais pas un seul article, donc inintéressant pour l'indexation.

Nous allons maintenant voir comment récupérer un article dans une page html.

4.1.2. Indexer le texte de l'article

Jusqu'à maintenant, nous avons pu identifier quelles sont les pages d'un site de presse qui contiennent un article. Le but, désormais est d'isoler l'article, car c'est lui qui nous intéresse. Comme pour l'indexation d'un titre, plusieurs stratégies s'offrent à nous pour l'indexation d'un article. En général, les pages contenants un article s'organisent suivant ce schéma :

Page contenant un article



La première piste pour indexer un article, était de se servir des tableaux dans le code html. En effet, tous les éléments de la page : menu, rubriques, article, publicités, charte sont

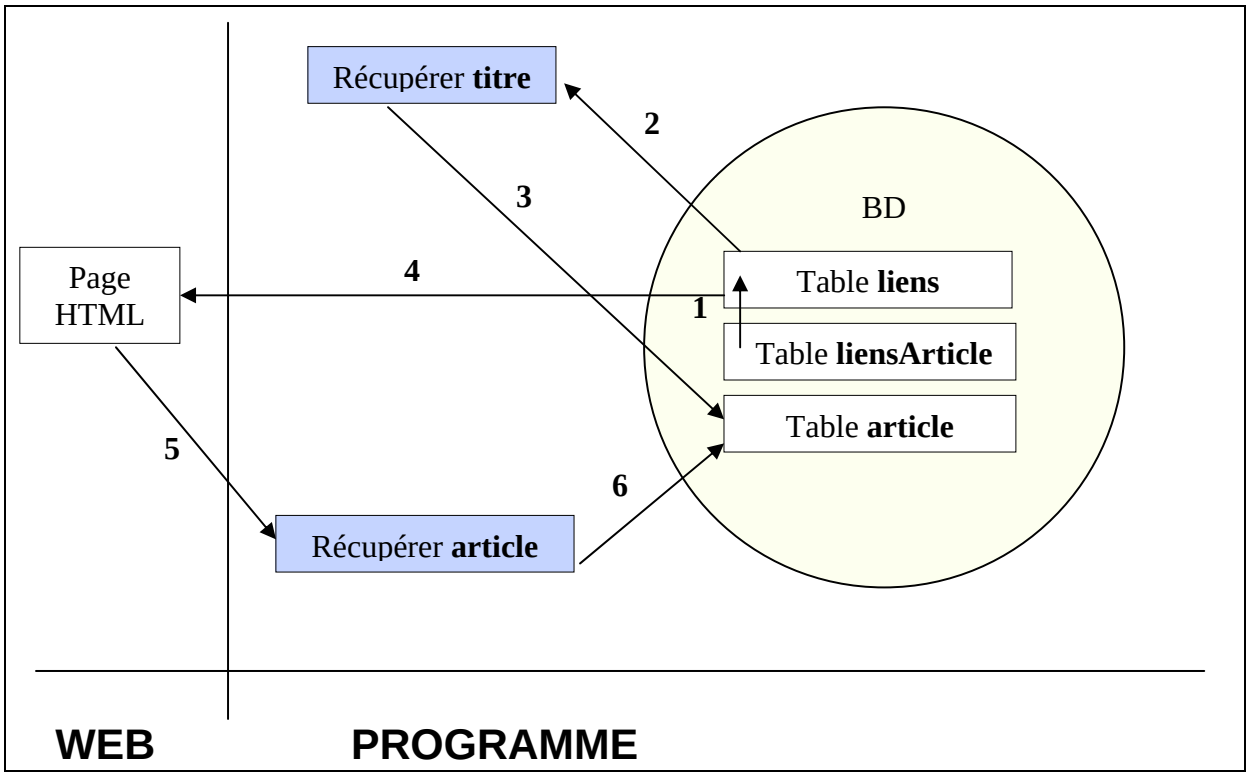
contenus dans des tableaux html (par les balises *<table>* pour le début du tableau, *<tr>* pour le début d'une ligne, *<td>* pour le début d'une cellule et leurs fermetures correspondantes *</td>*, *</tr>* et *</table>*). Le but était donc d'arriver à enlever tous les tableaux html sauf celui du milieu contenant l'article. Seulement, cette solution n'est pas standard pour tous les sites, car malgré une structure commune au niveau de l'apparence, dans le code html ils sont tous différents. De plus, si les balises html de tableaux ne sont pas toutes fermées ou qu'elles s'entrecroisent, appliquer un algorithme permettant de ne récupérer que le contenu d'un seul tableau (celui de l'article) ne marcherait pas pour tous les sites de presse. Nous avons donc opté pour une autre stratégie qui est de ne garder que ce qui est texte dans la page. C'est à dire qu'il y aura beaucoup d'éléments à enlever dans le code html, chose qui se fera grâce aux expressions régulières.

La première chose à faire est de ne garder que le corps du document, c'est à dire enlever toutes les en-têtes. Plus précisément on retire tout ce qui est avant la balise html *<body>* et tout ce qui est après la balise *</body>*. Ensuite, on supprime tous les liens. En fait, on enlève tout ce qui y a entre les balises de lien *<a>* et **, donc les textes de liens, ainsi que ce qui pourrait y avoir entre les balises *<option>* et *</option>* et les balises *<script>* et *</script>*. C'est à dire les labels de formulaires et le JavaScript. Dans le but d'avoir un texte vraiment propre, on appliquera ensuite à la chaîne (le code de la page) un débalisage complet : toutes les balises html (et autres tags) seront enlevées. Pour finir, on effectue un nettoyage complet de la chaîne de caractères :

- Le titre de l'article, souvent répété, sera supprimé à chaque occurrence de celui-ci.
- On peut supprimer aussi quelques caractères spéciaux comme ** ** (tabulation)…
- Tout ce qui est sauts de ligne et espaces (avant le début du premier mot de la chaîne et après la fin du dernier mot de la chaîne) seront supprimés.

Finalement, on se retrouve avec une chaîne de caractères nettoyée, et surtout correspondant à l'article que l'on veut isoler.

On va maintenant voir, d'un point de vue général, le principe de l'algorithme d'indexation d'un article et de son titre, ainsi que l'interaction avec les tables de la base de données. Pour chaque lien considéré comme pointant vers un article (tous les liens identifiés dans la table **liensArticle**), on récupère son titre (grâce à la table **liens**) et on récupère l'article correspondant au lien qui le pointe, puis on insère l'article avec un identifiant unique (récupéré dans la table **liensArticle**) ainsi que son titre correspondant dans la table **article**. Lors de l'insertion dans la base de données, on n'oubliera pas de protéger les caractères de guillemets et d'apostrophes des articles et des titres pour éviter des incohérences et des erreurs. Le schéma suivant reprend les grandes lignes de l'algorithme que l'on vient de voir :

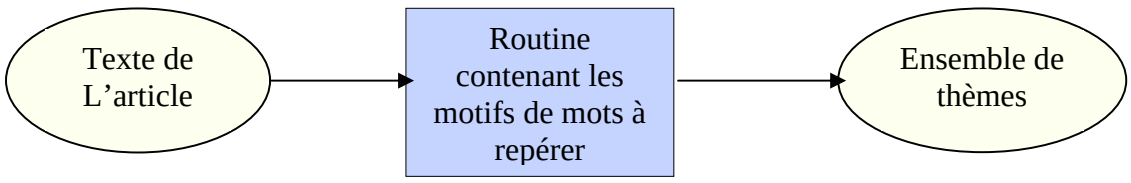


Les chiffres sur le schéma correspondent aux instants dans le temps. Cela montre bien que les actions « récupérer le titre » et « récupérer l'article » se déroulent séquentiellement. Alors que les actions « insérer le titre dans la table **article** » et « insérer l'article dans la table **article** » se déroulent en même temps. L'algorithme met à peu près 15 minutes pour récupérer les articles et leurs titres (pour le site de presse: Ouest-France.fr).

4.2. La stratégie à adopter pour indexer les termes

4.2.1. Les motifs de mot utilisés

Nous devons à présent récupérer les thèmes d'un article. Les thèmes (ou termes) d'un article de presse sont en fait les mots ou groupes de mots qui ont une occurrence supérieure à un dans l'article. Par ailleurs, nous avons décidé de prendre comme thèmes seulement les groupes de mots (au moins deux mots dans un terme) dans l'intérêt de n'avoir que des thèmes plus parlants et plus intéressants pour l'utilisateur. Dans le but de repérer des thèmes possibles dans un article, on passe le texte de l'article dans une routine, qui contient des motifs de mots à repérer, puis elle nous ressort tous les termes correspondants à ces motifs. Ces motifs seront implémentés dans la routine par des expressions régulières. Voici le schéma du principe de récupération des thèmes (ensemble de termes) :



Maintenant nous allons voir un peu plus précisément les motifs que nous avons utilisés. Nous avons tout d'abord les motifs primaires qui serviront à repérer les motifs de mots qui nous intéressent :

- **Maj** : tous les mots dont le premier caractère est une lettre majuscule et les suivantes sont soit majuscules, soit minuscules.
- **Min** : tous les mots de trois lettres ou moins n'ayant que des lettres minuscules.
- **Spa** : tout caractère d'espace.
- **Ponct** : tout caractère de ponctuation.
- **MajLetter** : une seule lettre majuscule.

Voyons maintenant les motifs de mots qui nous permettront d'isoler les thèmes d'un article :

- **Maj.Spa.Min.Spa.Maj** : permet de récupérer des groupes de mots comme par exemple XV de France, Guerre en Irak. De plus on voit bien que le mot en minuscule est un mot vide et qu'il est entre deux mots pleins.
- **Maj.Spa.Maj** : permet de récupérer deux mots pleins qui se suivent et qui sont de motif Maj tous les deux. Par exemple : Jacques Chirac.
- **Maj.Spa.MajLetter.Ponct.Spa.Maj** : permet de récupérer des termes comme par exemple Georges W. Bush.
- **Maj.Spa.Min.Ponct.Maj** : est presque similaire au premier sauf que le mot vide représenté par Min dans le premier motif est remplacé par la concaténation des motifs primaires Min.Ponct, qui permettent de récupérer des termes comme par exemple Côte d'Ivoire.
- **Maj.Ponct.Maj.Spa.Maj** : permet de récupérer des termes comme par exemple Jean-Pierre Raffarin.
- Il y a aussi d'autres motifs qui sont assez similaires à ceux déjà cités…

Ce qu'il faut voir, c'est que nous avons opté pour une stratégie particulière afin de récupérer les thèmes d'un article. Le principe de base est de ne prendre que les groupes de mots commençant par une lettre majuscule (ou étant entièrement en majuscules) en fonction de motifs déjà déterminés. La plupart des mots en majuscules, contenus dans ces groupes de mots, dans leur contexte, sont considérés comme des mots pleins (importants, qui définissent le thème). On se sert des mots vides (peu importants quand ils sont tous seuls, mais importants et intéressants quand ils sont en relation avec des mots pleins) dans certains motifs dans le but de récupérer un terme plus proprement et plus significatif. En effet, le terme **Guerre en Irak**, est composé de deux mots pleins que sont **Guerre** et **Irak**, et d'un mot vide qui est **en**. Le mot vide tout seul ne veut rien dire, mais dans le terme il situe où se trouve la Guerre (en Irak). D'où l'importance de créer des motifs de mots et de bien composer ceux-ci en fonction de mots pleins et de mots vides.

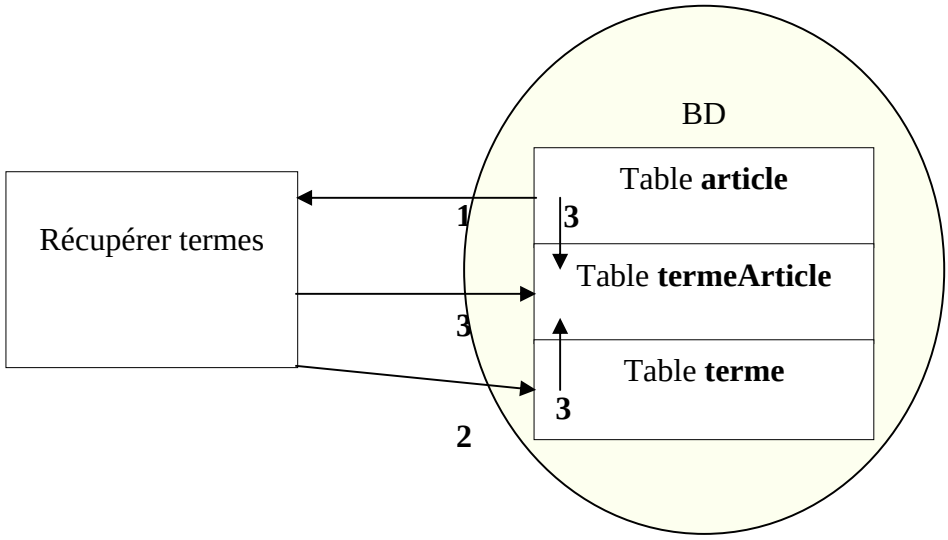
Nous allons voir maintenant l'utilisation de ces motifs de mots, l'indexation des termes et leurs relations avec les articles.

4.2.2. Le traitement des mots et l'insertion dans la base de données

Le principe général de ce qui reste à faire est qu'à partir des articles qui sont indexés dans la base de données, on récupère tous les termes et on les indexe à leur tour dans la base. On se base donc sur l'algorithme suivant :

```
pour tout article étant indexé dans la table article,
    on applique la routine contenant les motifs de récupérations de termes
    on teste pour chaque terme s'il existe déjà dans la table terme
    Si c'est le cas,
         on insére les bons identifiants dans la table termeArticle
    Sinon,
         on insère le terme dans la table terme
         on indique sa relation avec l'article dans termeArticle
fin
```

Voici le schéma qui décrit l'algorithme précédent :

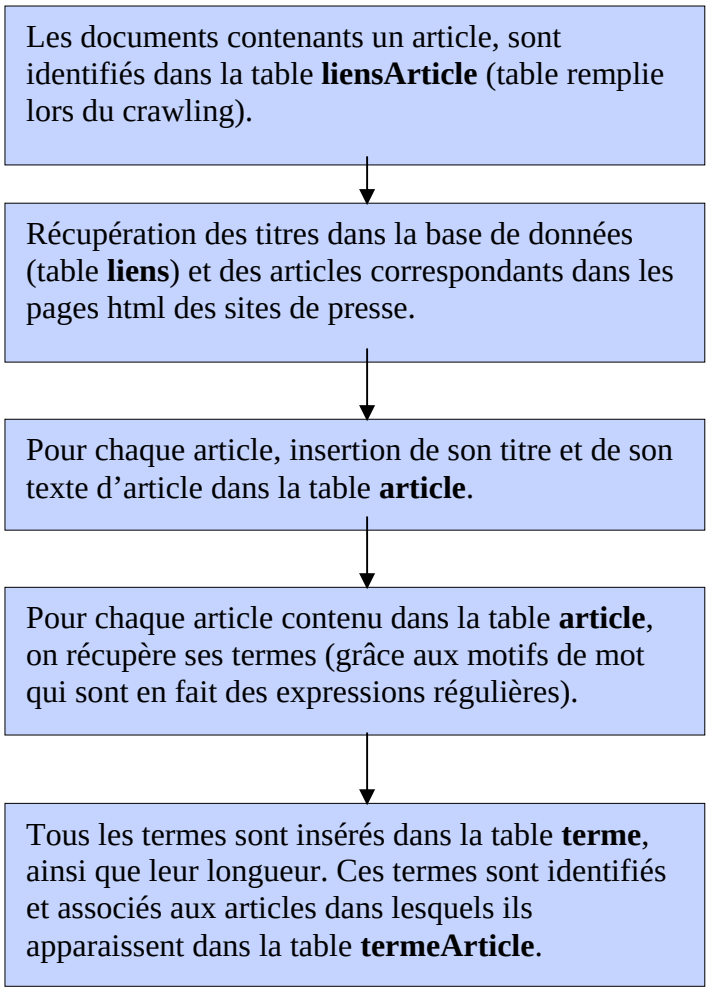


Comme pour le schéma d'indexation des articles, les chiffres correspondent aux instants dans le temps. L'algorithme s'exécute en 10 minutes à peu près pour indexer tous les termes (pour le site de presse: Ouest-France.fr). On voit bien la création d'une relation entre un terme et un article en associant dans la table **termeArticle**, l'identifiant du terme et celui de l'article dans lequel le terme a été trouvé. Nous avons choisi d'établir cette table dans la base de données dans le but de pouvoir gérer le fait d'avoir un terme associé à plusieurs articles et un article associé à plusieurs termes. De plus cela permet d'étudier le croisement des termes dans les articles, pour faciliter la navigation de termes en termes de l'utilisateur. Ou bien dans le but d'obtenir des statistiques entre les articles et leurs termes.

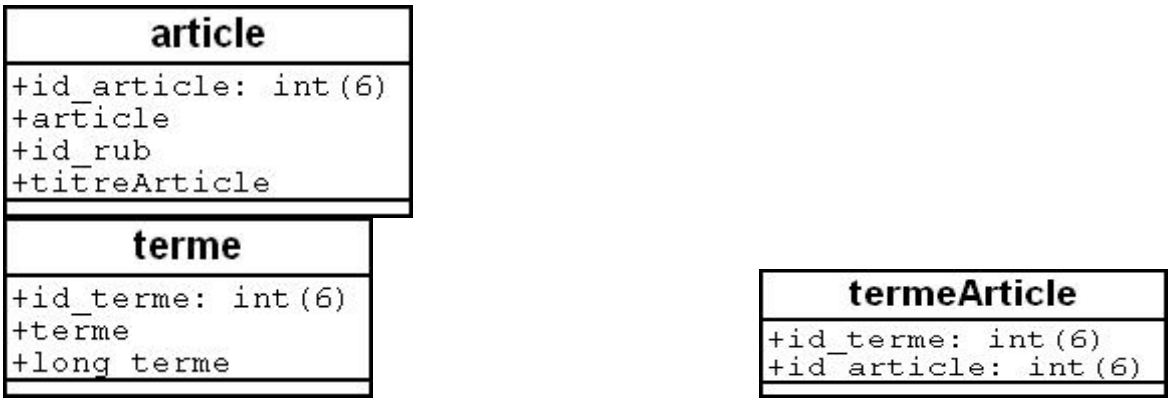
Avant l'insertion des termes dans la table **terme**, tout comme l'insertion des articles, on doit protéger les caractères de guillemets et d'apostrophes contenus dans ces termes. En outre, pour chaque terme inséré, on calcule sa longueur (le nombre de caractères dans la

chaîne) que l'on insère aussi dans la table **terme**. La longueur du terme aura pour but de sélectionner les termes à afficher dans l'interface.

Dans les grandes lignes, l'indexation des articles et des termes d'un site de presse peut se schématiser ainsi :



Toutes les informations concernant le site de presse (l'article, son titre, et ses termes), sont comme nous l'avons dit, insérées dans une base de données dont voici le détail des tables qui concernent l'indexation :



L'enregistrement des données des sites de presses est maintenant achevé, nous allons pouvoir traiter l'interface graphique et la présentation de l'information.

5. La conception de l'interface, choix

5.1. Création d'une architecture

La première chose à faire est de définir un type d'interface, un squelette du site qui sera statique dans un premier temps, statique dans le sens où il n'y a pas de base de données derrière pour rendre le site dynamique. Nous allons vous expliquer les techniques que nous avons utilisé pour présenter l'information.

L'architecture de notre interface est construite de la façon suivante :

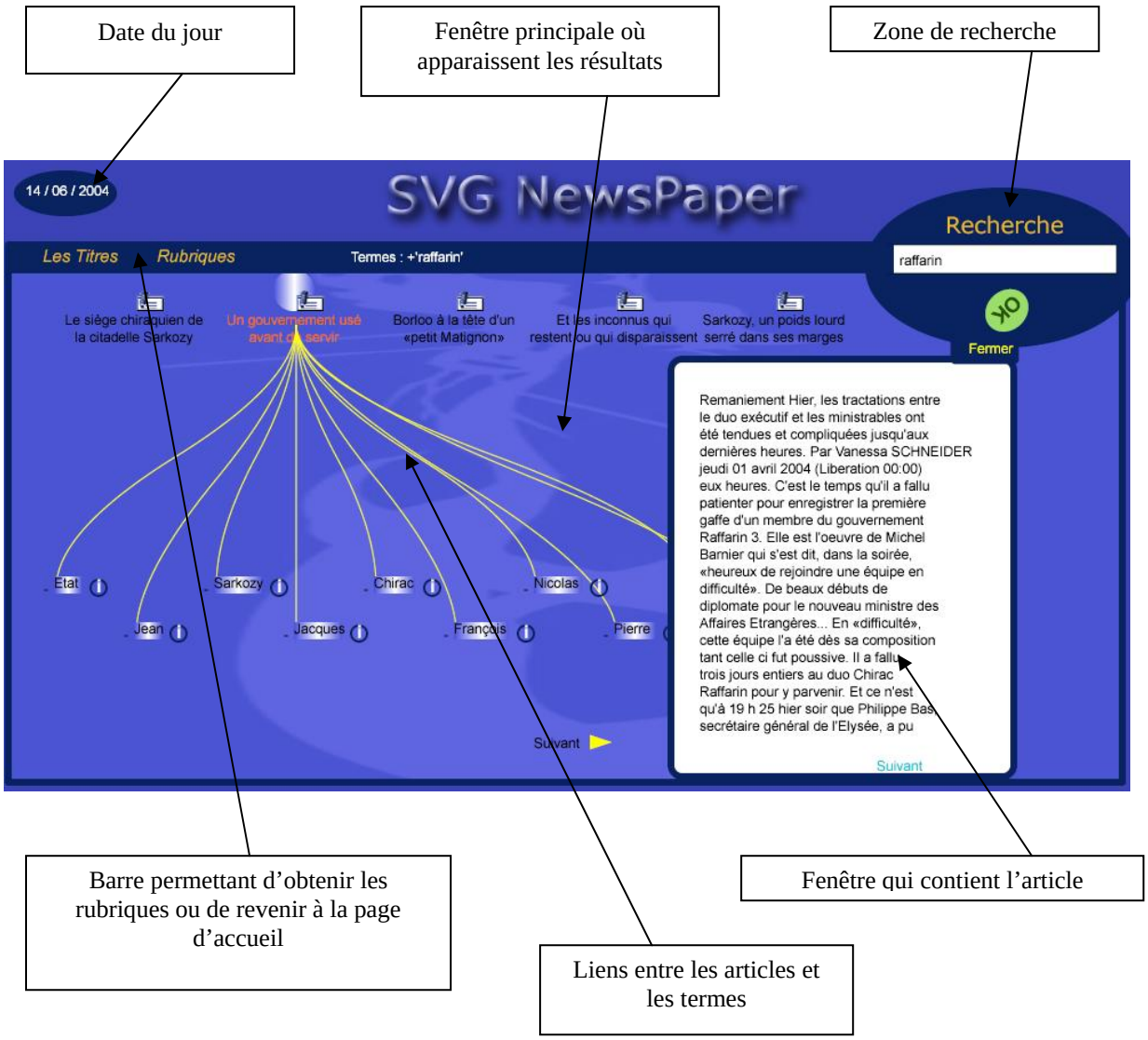


Fig.8. Description de notre interface

Nous avons réfléchi sur plusieurs types d'interfaces avec plusieurs rectangles (un pour les rubriques, un pour la recherche, un pour les résultats, un pour les articles…) mais cela ne convenait pas car la place pour afficher les résultats de la recherche se trouvait réduite. Nous avons donc décidé de créer une interface qui privilégie les résultats. Les autres éléments (contenu de l'article, statistiques…) n'apparaissent que lorsque l'utilisateur le décide (chaque élément apparaît par mouvement de translation).

- ✓ Système de coordonnées

Lors de la création d'une interface en SVG, nous travaillons beaucoup avec les coordonnées des objets. Le coin supérieur gauche de la fenêtre correspond à l'origine du repère. L'axe des abscisses est dirigé de gauche à droite et celui des ordonnées de haut en bas.

Tous les éléments doivent être affichés avec leurs positions. Le texte fait partie des éléments graphiques et donc nous avons du faire une fonction pour gérer le retour à la ligne. Il n'existe pas de marqueur de retour à la ligne en SVG comme on peut le trouver en HTML. La fonction va mettre le texte à la ligne toutes les « n » longueurs. Nous comptons la longueur de chaque mot puis nous incrémentons un compteur. Si le compteur arrive à « n » alors on passe à la ligne. Si le compteur est supérieur à « n » avec le dernier mot alors on passe à la ligne et on affiche ce mot puis on relance la même procédure avec le reste du texte. On évite ainsi de couper les mots.

✓ Principe de visualisation des articles

Nous avons choisi d'éviter l'utilisation de la scrollbar dans les articles et nous avons choisi de garder le principe des livres : tourner une page.

On a alors à la fin de chaque page d'un article un bouton « précédent » et un bouton « suivant ».

Fig.9. La fenêtre de l'article

Pour mettre en place ce type de visualisation nous comptons le nombre de lignes que nous affichons puis à partir d'un nombre de lignes donné nous créons une nouvelle division (au sens du DOM de la W3C) qui va permettre de créer la page suivante de l'article.

Une fois les divisions créer nous les rendons invisibles en modifiant l'attribut de chaque division. Quand l'article apparaît nous mettons la première division visible. Quand on fait une transition d'une division à une autre on appelle une fonction cacheTout() qui permet de cacher toutes les divisions puis ensuite on rend visible la division souhaitée.

Par exemple si on a découpé l'article 1 en 3 pages nous aurons les divisions :
«1_0 », «1_1 » et «1_2 »

Fonction *alaligne* qui permet de mettre le texte à la ligne et de générer les pages des articles :

```
function alaligne($string,$numero){ //$string= l'article, $numero : numero de l'article
  $cptligne=0;
  $tok = strtok($string," ");
  $cpt=220;
  $chaine="";
  $first=1;
  $num=0;
  while ($tok){
    if(($cptligne==0)&&($first==1)){
      $num2=$numero."_".$num;
      echo"<g id=\"$num2\" visibility=\"hidden\">\n";
    }
    $first++;
    if((strlen($chaine)+strlen($tok))>40){
      echo"<text x=\"1125\" y=\"$cpt\">$chaine</text>";
      $cptligne++;
      $chaine="";
      $cpt+=15;
    }
    $chaine.=" ".$tok;
    $tok = strtok(" ");
     if($cptligne==20){
      $cpt+=20;
      if($num==0){
          $t=$num+1;
          echo"<a><text x=\"1300\" y=\"540\" style=\"text-anchor: middle; fill:#22C3D1\"
 onclick=\"montre(evt, '$t','$numero','1')\"> Suivant </text></a></g>\n";
      }
      else{
          $t=$num-1;
          echo" <a><text x=\"1200\" y=\"540\" style=\"text-anchor: middle; fill:#22C3D1\"
 onclick=\"montre(evt, '$t','$numero','0')\"> Précédent </text></a>";
          $t=$num+1;
          echo" <a><text x=\"1300\" y=\"540\" style=\"text-anchor: middle; fill:#22C3D1\"
 onclick=\"montre(evt, '$t','$numero','1')\"> Suivant </text></a></g>\n";
      }
      $cptligne=0;
      $first=1;
      $num++;
      $cpt=220;
    }
  }
  $cpt+=20;
  $n=$num-1;
  if($num!=0){
    echo" <text x=\"1200\" y=\"540\" style=\"text-anchor: middle; fill:#22C3D1\"
 onclick=\"montre(evt,'$n','$numero','0')\"> Précédent </text></g>";
```

```
  }
  else{
    echo"</g>";
  }
  return $num;
}
```

- ✓ Affichage des articles

Pour faire apparaître la fenêtre des articles, nous utilisons la translation. Nous créons cette fenêtre hors de l'interface principale (avec des coordonnées négatives), de façon à ce qu'elle ne soit pas visible à l'écran. Ensuite, lorsque l'utilisateur clique sur le bouton qui permet d'afficher l'article, nous effectuons la translation, c'est-à-dire que les coordonnées de la fenêtre changent pour se retrouver à l'intérieur de l'écran.

Nous avons utilisé aussi cette technique pour les rubriques, pour afficher les informations sur les personnes qui ont participé à ce projet, ainsi que pour faire apparaître les statistiques concernant les occurrences d'un terme dans un article.

Techniquement nous utilisons les balises de SVG pour déclencher une animation simple, par un seul événement.

Exemple :

```
<animateTransform id="animFenetre" attributeName="transform" type="translate" from="-200" to="300" begin="0s" dur="1s" fill="freeze" />
```

Si cette animation doit se déclencher sur plusieurs événements nous utiliserons un script EcmaScript. Ce script est détaillé dans la partie « A la découverte de SVG/Animations et scripting).

- ✓ Création des liens

Pour créer les liens entre les termes et les articles, nous avons utilisé les courbes de Bézier. Ces courbes sont définies par 4 points :

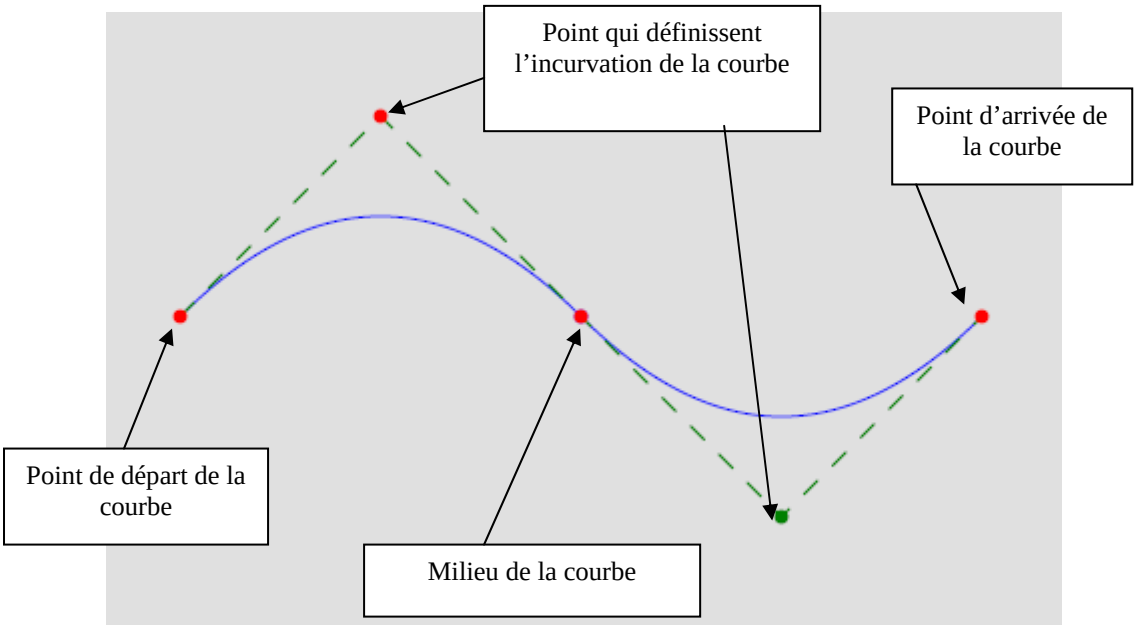


Fig.10. Principe des courbes de Bezier

SVG nous permet de définir des courbes de Bézier à l'aide de la balise <PATH>.

Bien évidement, il a fallu trouver une solution mathématiques pour que la courbe fluctue de part et d'autre de la droite définie par point de départ (article) – point d'arrivée (terme) mais sans trop s'en écarter. Nous avons utilisés des courbes de Bézier.

On définit un tracé avec un élément *path* ayant un attribut *d = "<données-de-tracé>"*, où *d* contient les commandes *moveto*, *line*, *curve* (avec des courbes de Bézier cubique ou quadratique), *arc* et *closepath*.

- ✓ Les commandes "moveto" (M ou m) établissent un nouveau point courant

- ✓ Les commandes de courbe de Bézier quadratique (Q ou q) dessinent une courbe de Bézier quadratique, du point courant jusqu'au point (x,y), en utilisant le point (x1,y1) comme point de contrôle. On peut spécifier plusieurs jeux de coordonnées pour le dessin d'un polybézier. Une fois la commande exécutée, le nouveau point courant devient la paire de coordonnées finale (x,y) utilisée dans le polybézier.

- ✓ Les commandes de courbe de Bézier quadratique (T ou t) dessine une courbe de Bézier quadratique, du point courant jusqu'au point (x,y). Le point de contrôle est sensé être le reflet du point de contrôle de la commande précédente par rapport au point de contrôle. (S'il n'y a pas de commande précédente ou si celle-ci n'était pas une Q, q, T ou t, le point de contrôle est sensé coïncider avec le point courant).Une fois la commande exécutée, le nouveau point courant devient la paire de coordonnées finale (x,y) utilisée dans le polybézier.

Exemple :

<path d="M340,380 Q340,330,370,265,T400,150" style="stroke:#4D54DE;fill:none"/>

Fig.11. Rendu SVG du code ci-dessus

- ✓ Zone de recherche

Les champs de formulaire n'existe pas en SVG, nous avons donc du en créer. Nous avons en fait dessiné un rectangle avec une zone de texte vide. Cette étape relativement facile ne permet à l'utilisateur de rentrer son texte.

Il a donc fallu créer un script qui va être déclenché quand l'utilisateur va cliquer sur le rectangle qui désigne le champ et ainsi il pourra rentrer son texte.

On va reconnaître les frappes de l'utilisateur, et ainsi nous allons pouvoir inscrire ce qu'il recherche dans le champ en modifiant le contenu de la zone de texte.

Code de la fonction :

```
var phrase="";
function tape(evt){
        svgdoc=evt.getTarget().getOwnerDocument();
        key = evt.getCharCode();
        if ((key==8)&&(phrase!="")) {
            phrase=phrase.substring(0,phrase.length-1)
              }
        else{
            lettre = String.fromCharCode(key);
            phrase=phrase+lettre

       }
            cible="champ_recherche"
           objet=svgdoc.getElementById(cible);
            child=objet.getFirstChild();
            child.setData(phrase)
           }
```

Cette étape passée il a fallu récupérer le résultat du champ quand l'utilisateur va lancer sa recherche via le bouton « ok ».

Pour ce faire nous accédons au contenu de la zone de texte puis nous récupérons la donnée (Chaque élément en SVG est accessible si nous respectons le D.O.M). Une fois la donnée récupérée nous appelons l'URL de notre page avec les nouveaux paramètres de la recherche.

Code de la fonction :

```
function getTerme(evt){
                svgdoc=evt.getTarget().getOwnerDocument();
                cible="champ_recherche";
                objet=svgdoc.getElementById(cible);
                child=objet.getFirstChild();
                t= child.getData();
                var z= "<?php echo $_SERVER["PHP_SELF"]; ?>";
                window.location.href = z+"?terme="+t;
}
```

Nous avons créé aussi une fonction pour remettre le champ vierge qui est appelé quand l'utilisateur va commencer une nouvelle saisie.

Notre site a déjà bien avancé, il va falloir maintenant travailler sur le contenu du site et le tri de l'information qui est stocké dans la base de données.

- ✓ Création des graphiques

Nous avons voulu représenter des informations complémentaires sur les termes. Nous avons décidé de noter la proportion d'un terme dans différents articles.

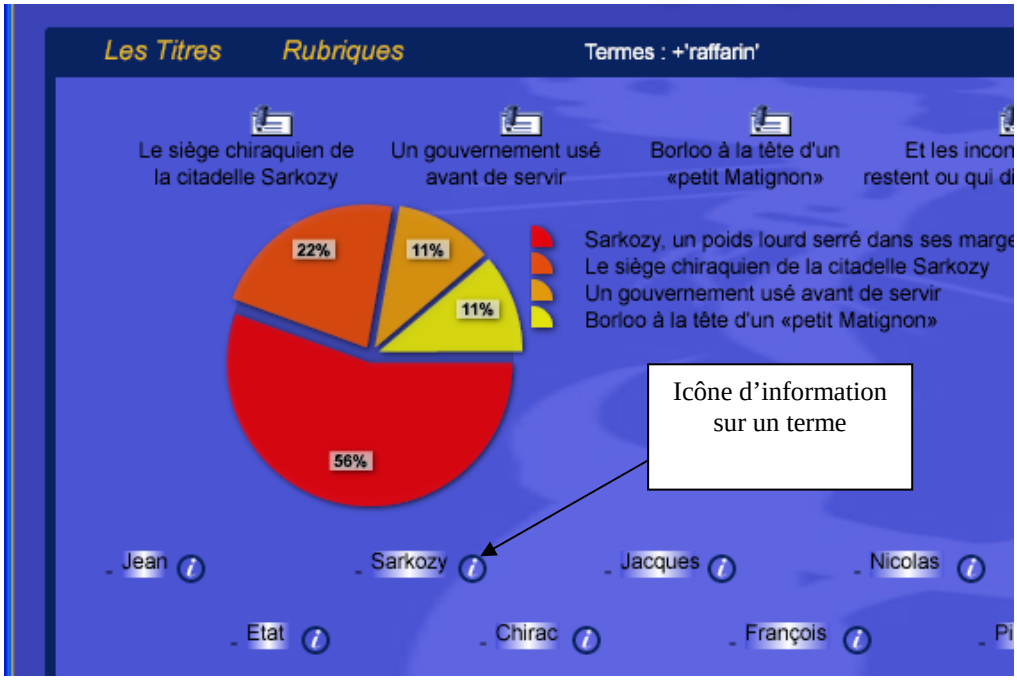


Fig.12. Information sur les termes

A ce stade nous n'avons pas les données donc nous avons créé une fonction qui va prendre en paramètre un tableau de valeur et un tableau de noms.

La fonction *makeGraphe* prend aussi le nom du terme et son identifiant.

Nous dessinons en fait un cercle vide puis pour chaque valeur du tableau de données nous allons dessiner la part qu'il représente dans le cercle. Nous utilisons la commande *path* de SVG qui gère les tracés.

Partie principale de la fonction qui permet de dessiner les parts du graphique :

```
for ($i=0;$i < sizeof($graph_data); $i++) {
                        echo "<path d=\"M 0 0 l +100 0 A 100 100 0 ";
                        if ($graph_angles[$i]>$pi) {
                           echo "1 1";
                                }
                        else {
                            echo "0 1";
                                }
                echo " ". round(100*cos($graph_angles[$i])." ".
                round(100*sin($graph_angles[$i])." Z\" ";
                echo "style=\"fill:".$graph_colors[$i].";fill-opacity:.7\" ";
                echo"transform=\"translate(".round(6*cos($old+($graph_angles[$i]/2)))."
                 ".round(6*sin($old+($graph_angles[$i]/2))).")
                 rotate(".round($old/$pi*180).")\"/>";
                  $old+=$graph_angles[$i];
        }
```

Ensuite il nous a fallu marquer la légende sur le coté du graphique avec le bon code de couleurs. Nous avons parcouru le tableau des noms puis nous les avons écrits.

5.2. Requête sur les termes et articles

Après avoir défini notre modèle d'interface pendant que l'autre équipe s'occupait du crawling, il nous a fallu définir les requêtes utiles à nos besoins pour une présentation de l'information la plus pertinente possible.

Il a fallu recroiser les différentes tables de notre base pour pouvoir trier l'information.

Le premier travail s'est situé sur les termes et sur la page de présentation (les termes du jour). Nous avons défini un mode de calcul pour choisir les termes les plus pertinents : nous faisons le produit de son nombre d'occurrences par sa longueur. En effet nous avons pu voir que les mots courts sont les mots les moins significatifs pour l'apport d'information.

Exemple de requête SQL :

```
mysql_query("
SELECT terme.terme AS terme, terme.id_terme AS id_terme, terme.lon AS
longueur, COUNT(terme.id_terme) AS nb
FROM  termearticle, terme
WHERE termearticle.id_terme=terme.id_terme
GROUP  BY id_terme
ORDER BY nb DESC, longueur DESC LIMIT 10");
```

Ensuite nous avons travaillé sur les requêtes où l'utilisateur va rentrer un terme ou un ensemble de termes. Ici il a fallu recouper les termes avec les articles.

Nous recherchons d'abord les articles qui contiennent les termes recherchés puis nous ordonnons les articles par nombre d'occurrences des termes de ces derniers de manière décroissante. Nous avons donc obtenu les articles les plus pertinents pour l'utilisateur.

Les identifiants des articles vont être stockés dans une variable que l'on va construire et qui sera une variable qui respectera la syntaxe SQL pour qu'elle puisse être introduite dans une autre requête.

Requête SQL :

```
$requete = mysql_query("
SELECT article.id_article AS numarticle, article.titreArticle AS titre, COUNT(terme.id_terme) as nb
FROM terme, termearticle, article
WHERE  terme.id_terme=termearticle.id_terme and termearticle.id_article = article.id_article and terme.terme='".$_GET[terme]."'
GROUP BY numarticle
ORDER BY nb DESC");
```

Code pour la création de la variable :

```
$liste_article="(";
while(($rownb<$offset+$nbmax)&&($ligne =mysql_fetch_array($requete))){
  $tabIdArticle[$i]=$ligne[numarticle];
  if ($rownb>=$offset){
   if($rownb==$offset){
      $liste_article.=$ligne[numarticle];
      }
      else{
        $liste_article.=",".$ligne[numarticle];  ←Stockage des articles
      }
}
$liste_article.=")";
```

Par exemple à la fin de la boucle on se retrouve avec *$liste_article='' (1, 5, 112, 34)'';*

Maintenant il nous faut trouver les autres termes qui relient ces articles entre eux pour pouvoir faire naviguer l'internaute dans les graphes de termes.

Pour se faire, nous avons stocké les numéros des articles que l'on a triés puis on va les utiliser dans une nouvelle requête : cette requête va choisir tous les termes en commun aux articles choisis puis va les classer par nombre d'occurrences de manière décroissante. Ici on se limite au dix meilleurs termes.

La requête SQL :

```
$requete = mysql_query("
SELECT terme.id_terme AS idTerme, terme.terme AS nomterme, COUNT(terme.id_terme) AS nb
FROM termearticle, terme where termearticle.id_article IN $liste_article AND termearticle.id_terme=terme.id_terme AND terme.terme!='".$_GET[terme]."'
GROUP BY terme.id_terme
ORDER BY nb DESC LIMIT 10");
```

5.3. Vers un site dynamique, insertion des requêtes

Nous avons maintenant notre architecture et nos requêtes, nous devons maintenant combiner les deux pour que notre site fonctionne correctement.

Cette étape appelle les fonctions définies dans l'architecture du site (fonction *alaligne*…) avec en paramètres les résultats des requêtes SQL.

On appelle les fonctions dans des boucles : tant que la requête donne des enregistrements alors appliquer les différentes fonctions aux enregistrements.

Exemple :

```
while($ligne=mysql_fetch_array($requete)){
$terme=$ligne[terme];
  if($ligne[idterme]!=$idterme){
     if($idterme!=""){
       makegraph($tabdata, $tabnom, $ligne[terme], $idterme);
     }
…
}
```

5.4. Finalisation

Cette phase nous a permis de croiser nos résultats avec les requêtes SQL simples tapées dans une fenêtre MySQL pour vérifier si l'on n'avait pas perdu d'information quand on a travaillé avec nos tableaux et nos différentes fonctions.

Ensuite on a pu retoucher quelques effets visuels sur l'interface tel que des effets sur les textes au passage de la souris…

Dans cette période nous avons aussi fait « goûter » notre site à un public le plus large possible pour avoir un retour sur l'interface.

6. Les outils utilisés pour l'interface

6.1. Le dynamisme du site avec PHP et MySQL

PHP et MySQL forme le couple idéal pour générer un contenu dynamique dans un site web. PHP va permettre de faire passer des variables de pages en pages, de travailler sur du texte, de manipuler des tableaux… MySQL va permettre d'interroger notre base de données et ainsi créer nos requêtes.

Nos choix se sont orientés vers ces langages pour différentes raisons :
-Leurs portabilités.
-La syntaxe de PHP qui se rapproche des langages tel que le C.
-PHP est un langage procédural avec possibilité de faire de l'objet.
-PHP et MySQL sont des langages libres.
-PHP s'exécute coté serveur ce qui est pratique pour la sécurité.

6.2. A la découverte de SVG

SVG signifie Scalable Vector Graphics. Il s'agit d'une recommandation du W3C (World Wide Web Consortium), basée sur XML, pour les graphiques vectoriels en deux dimensions. Ce format supporte également les animations, comme en témoigne notre site.

SVG admet trois types d'objets graphiques : des contours graphiques vectoriels (par exemple, des tracés consistant en lignes droites et courbes), des images et du texte. Les objets graphiques peuvent être regroupés, stylés, transformés et composés dans des objets précédemment rendus. L'ensemble de fonctions comprend des transformations imbriquées, des tracés de rognage, des masques basés sur la couche alpha et des objets de gabarit.

Les dessins SVG peuvent être interactifs et dynamiques. Des applications sophistiquées de SVG sont réalisables avec le renfort d'un langage de script pour agir sur le Modèle Objet du Document (DOM) de SVG qui offre un accès complet à tous les éléments, attributs et propriétés. Nous avons choisit le langage EcmaScript qui est très proche du Javascript.

Un large jeu de gestionnaires d'événements, tel que *onmouseover* et *onclick*, peut être assignés à tout objet graphique SVG.

Comme SVG s'appuie sur le XML, nous allons vous présenter le XML puis ensuite quelques éléments particuliers et propres à SVG.

6.2.1. SVG basé sur XML

XML (eXtensible Markup Language, langage à balises extensibles) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises. La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

En réalité les balises XML décrivent le contenu plutôt que la présentation (contrairement à HTML). Ainsi, XML permet de séparer le contenu de la présentation ce qui permet par exemple d'afficher un même document sur des applications ou des périphériques

différents sans pour autant nécessiter de créer autant de versions du document que l'on nécessite de représentations.

- Mise en forme d'un document XML

XML est un format de description des données et non de leur représentation, comme c'est le cas avec HTML. La mise en page des données est assurée par un langage de mise en page tiers. A l'heure actuelle il existe trois solutions pour mettre en forme un document XML :

- ✓ CSS (Cascading StyleSheet), la solution la plus utilisée actuellement, étant donné qu'il s'agit d'un standard qui a déjà fait ses preuves avec HTML, que nous utiliserons.
- ✓ XSL (eXtensible StyleSheet Language), un langage de feuilles de style extensible développé spécialement pour XML. Toutefois, ce nouveau langage n'est pas reconnu pour l'instant comme un standard officiel
- ✓ XSLT (eXtensible StyleSheet Language Transformation). Il s'agit d'une recommandation W3C du 16 novembre 1999, permettant de transformer un document XML en document HTML accompagné de feuilles de style

- Structure d'un document XML

XML fournit un moyen de vérifier la syntaxe d'un document grâce aux DTD (Document Type Definition). Il s'agit d'un fichier décrivant la structure des documents y faisant référence grâce à un langage adapté. Ainsi un document XML doit suivre scrupuleusement les conventions de notation XML et peut éventuellement faire référence à une DTD décrivant l'imbrication des éléments possibles. Un document suivant les règles de XML est appelé document bien formé. Un document XML possédant une DTD et étant conforme à celle-ci est appelé document valide.

Exemple de DTD :

On définit un élément comme suit `<! ELEMENT Nom Modèle >` puis on peut assigner plusieurs types de Modèle :

Type prédéfini	Description
ANY	L'élément peut contenir tout type de données
EMPTY	L'élément ne contient pas de données spécifiques
#PCDATA	L'élément doit contenir une chaîne de caractère

Exemple de déclaration dans la DTD :

```
<! ELEMENT Nom (#PCDATA) >
```

Exemple d'utilisation :

```
<Nom>Dupond</Nom>
```

D'autre part il est possible de définir des règles d'utilisation, c'est-à-dire les éléments XML qu'un élément peut ou doit contenir. Cette syntaxe se fait à l'aide de notations spécifiques dont voici un récapitulatif:

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A **ou** B peuvent être présents (pas les deux)	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

Ainsi on peut créer la déclaration suivante dans la DTD:

```
<!ELEMENT personne (nom,prenom,telephone),email? >

<!ELEMENT nom (#PCDATA) >

<!ELEMENT prenom (#PCDATA) >

<!ELEMENT telephone (#PCDATA) >

<!ELEMENT email (#PCDATA) >
```

Cette déclaration pourra donc donner un document XML du style:

```
<personne>

        <nom>Dupond</nom>

        <prenom>Jean</prenom>

        <telephone>0231967867</telephone>

        <email>Dupond.J@wanadoo.fr</email>

</personne>
```

Bien d'autres éléments sont importants dans les DTD mais nous n'allons pas nous exposer plus dessus. On pourra voir les spécifications sur le site de référence : http://www.w3.org/TR/REC-xml#dt-doctype

- Décodage d'un document XML

XML permet donc de définir un format d'échange selon les besoins de l'utilisateur et offre des mécanismes pour vérifier la validité du document produit. Il est donc essentiel pour le receveur d'un document XML de pouvoir extraire les données du document. Cette opération est possible à l'aide d'un outil appelé analyseur (parseur).

Le parseur permet d'une part d'extraire les données d'un document XML ainsi que de vérifier éventuellement la validité du document.

- Avantages d'un document XML

La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML.

Auto descriptif et extensible.

Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques.

Universalité et portabilité : les différents jeux de caractères sont pris en compte.

Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP.

Intégrabilité : un document XML est utilisable par toute application pourvue d'un parseur.

Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

XML est particulièrement adapté à l'échange de données et de documents, donc SVG l'est aussi.

Notre choix de SVG s'est fait car il est facilement implémentable, on peut intégrer n'importe quel langage à l'intérieur du moment qu'il respecte les standards de la W3C. Il est gratuit contrairement à Flash de Macromédia. Il se marie très facilement avec Php qui permet de rendre notre site dynamique. Et surtout, il ne nous était pas connu.

6.2.2. Animation et scripting

Un des points forts de SVG c'est que l'on peut définir et déclencher des animations, soit déclarativement (i.e., en incorporant les éléments d'animation SVG dans un contenu SVG), soit via un script. Le principe pour les scripts est d'accéder à l'élément voulu puis de modifier ses attributs. On accède aux éléments par le biais du DOM (Document Object Model).

Exemple d'une animation pour faire glisser une fenêtre sans script :

```
<g id="fenetre">
<animateTransform id="animFenetre" attributeName="transform" type="translate" from="-200" to="300" begin="0s" dur="1s" fill="freeze" />
</g>
```

Cette animation va se déclencher au chargement de la page, on pourrait la déclencher sur un événement (onclick, onmouseover…) mais si on a une multitude d'éléments à déplacer il faudra alors répéter ce code pour chaque élément. On peut alors choisir de « scripter » l'animation.

Exemple de script pour faire glisser une fenêtre. Ici on peut rajouter des paramètres (comme le nom de l'objet à déplacer…) :

```
function glisse(evt, witch, from, to){
	objet=svgDocument.getElementById(witch);
	var X = objet.getAttribute('x')*1;
	if(X>to){
	}
	else{
		dx = vitesse * (to + X) /100;
		X += dx;
		objet.setAttribute('x', X);
		setTimeout("glisse(evt, which, from, to)", 25);
	}
}
```

Nous avons donc décrit l'ensemble du travail effectué et nous allons maintenant nous exprimer sur les difficultés rencontrées ainsi que notre opinion sur ce projet.

7. Difficultés rencontrées

Concernant le crawl :

- ✓ Faire en sorte que les navigateurs, et surtout le serveur web de la fac, n'arrête pas les fonctions au bout de 30 secondes. Nous avons résolu ce problème grâce à une fonction PHP (set_time_limit()) qui permet de dépasser les 30 secondes d'exécution pour une fonction.
- ✓ Trouver une méthode efficace pour différencier les articles des rubriques
- ✓ Optimiser le code du crawler un maximum pour ne pas passer 3 heures à crawler.

Concernant l'indexation :

- ✓ Bien comprendre la gestion et détection des groupes mots pleins / mots vides.
- ✓ Correctement nettoyer la page pour récupérer l'article

Concernant l'interface :

- ✓ Les formulaires

Trouver la manière de simuler un champ de formulaire HTML en SVG n'a pas été très facile et le résultat n'est pas aussi « propre » qu'un champ html car nous avons pas la possibilité de mettre l'indicateur de curseur dans le champ. C'est moins facile pour l'utilisateur de savoir où il en est.

- ✓ Les sessions en php

Au départ nous voulions utiliser les sessions php pour pouvoir faire un panier afin de sauvegarder les articles mémorisés pour que le lecteur puisse revenir sur sa sélection.

On souhaitait les utiliser aussi pour ne pas faire passer nos variables dans l'URL et ainsi éviter que les paramètres soient visibles à la manière de la méthode GET dans les formulaires HTML.

Nous avions créer le caddie virtuel mais au moment de l'intégrer dans notre code, il y a eu un conflit d'entêtes. Le navigateur ne reconnaît alors plus SVG. Ce problème vient du plugin. Nous avons essayé avec Davy Gigan de résoudre le problème mais sans résultat positif. La question a été posée sur une newsgroup PHP et SVG et ainsi que sur divers forum mais là aussi nous n'avons pas eu de solution.

- ✓ La base de données

Nous avons été obligé de transporter la base de données d'un serveur à un autre car on ne peut pas se connecter à la base de données de la fac via un autre serveur pour des raisons de sécurité. Nos scripts ont été placés sur un serveur personnel car nous n'avons pas pu travailler à la fac du fait qu'il y a eu des difficultés à installer un navigateur acceptant SVG.

8. Conclusion

8.1. Le travail en groupe

Ce travail en groupe de deux binômes était très intéressant et enrichissant. En effet, malgré quelques lenteurs de communications au début, nous avons réussi à bien nous répartir les tâches. Cette expérience était vraiment enrichissante car elle nous a appris à gérer notre temps et notre évolution dans le travail en fonction d'une autre équipe. L'une ne peut effectivement pas fonctionner sans l'autre.

8.2. Notre point de vue sur le projet

Ce projet était très intéressant, et de plus, il semble aller dans le sens de la plupart des webmasters : rendre la navigation sur les sites Internet plus facile. En effet, on voit déjà des sites comme le Monde ou Libération ajouter à chaque article des liens vers les articles ayant trait au même sujet.

Nous avons pu découvrir les coulisses des sites de presse, ce qui nous permet de mieux les connaître.

Nous avons pu travailler avec un nouveau langage, SVG, qui est très puissant (basé sur XML et on peut incorporer n'importe quelle langage à l'intérieur d'un code SVG qui respecte les standards XML) et en plein essor. Nous avons appris à découvrir ce langage par nous même et nous avons bien vu l'intérêt des newsgroups, forums et autres communautés de développeurs.

8.3. Améliorations possibles

Concernant le crawl nous aurions aimé traiter les points suivants :

- ✓ Possibilité de crawler le site sur toute sa profondeur. Cette méthode est très coûteuse en temps, mais permettrais une indexation totale d'un site.
- ✓ Possibilité de crawler un site avec une heuristique. La phase de crawl ne s'arrêterait alors qu'après avoir récupérer un nombre d'articles déterminés. Cette heuristique pourrait tenir compte de l'url absolues d'un fichier, par exemple l'arborescence des répertoires.
- ✓ Possibilité de mettre à jour des statistiques plus poussées pour chaque site. Ceci consisterait à mémoriser les rubriques, avec leurs chemins d'accès… Et si l'utilisateur affirme que ce n'est pas une rubrique, modifier la statistique.

Concernant l'indexation

- ✓ Respect des paragraphes, de la police de caractères…
- ✓ Ajout des images dans les articles

Concernant l'interface

Nous aurions aimé travailler avec les rubriques mais ces dernières sont difficiles à différencier lors de la partie crawling.

Nous aurions souhaité travailler avec les sessions en php pour faire notre caddie virtuel et éviter de faire passer les paramètres dans l'url.

Nous aurions souhaité intégrer les images dans les articles mais la récupération de ces dernières a échoué lors du crawl.

9. Références

Le langage SVG était nouveau pour nous. Nous avons dû l'apprivoiser. Pour ce faire nous avons utilisé plusieurs sites Internet. Nous nous sommes inscrits à la newsgroup en anglais traitant de SVG. Elle est très vivante.

9.1. SVG

-La spécification : http://www.w3c.org

-Tutorial et traduction de la spécification : http://www.yoyodesign.org/

- Exemples : http://pilat.free.fr/

- Exemples, cours : http://www.svgfr.org/

9.2. PHP/MySQL

-Manuel PHP : http://www.php.org

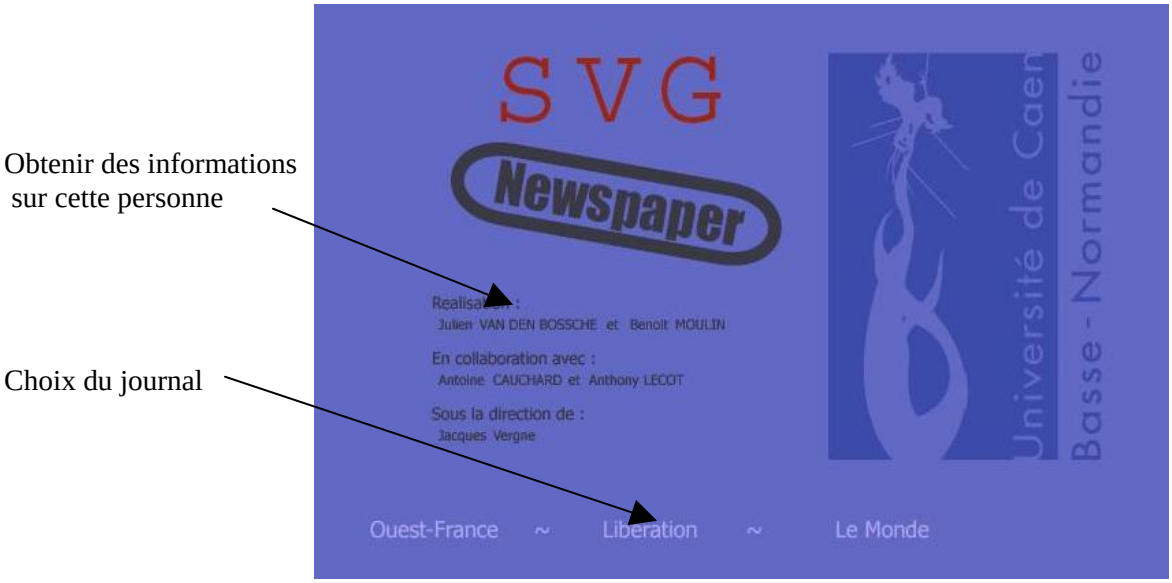
-Manuel MySQL : http://www.mysql.com

10. Annexes

Manuel d'utilisation

Choix du journal

Le choix du journal s'effectue sur la première page qui apparaît à l'écran.

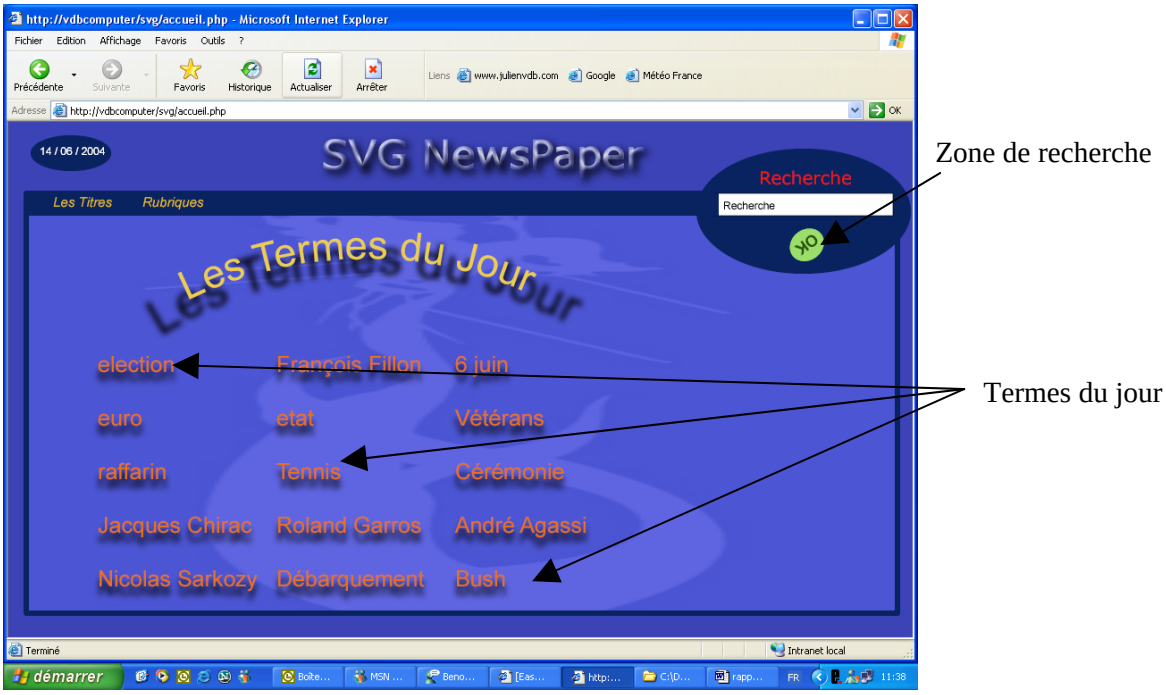


Pour choisir un journal, il suffit de cliquer sur son titre :

Cette première page permet également d'obtenir des informations sur les personnes qui ont participé à la réalisation de ce site. Pour cela, il faut cliquer sur le nom de la personne :

Recherche

Une fois le journal choisi, l'utilisateur peut lancer une recherche :

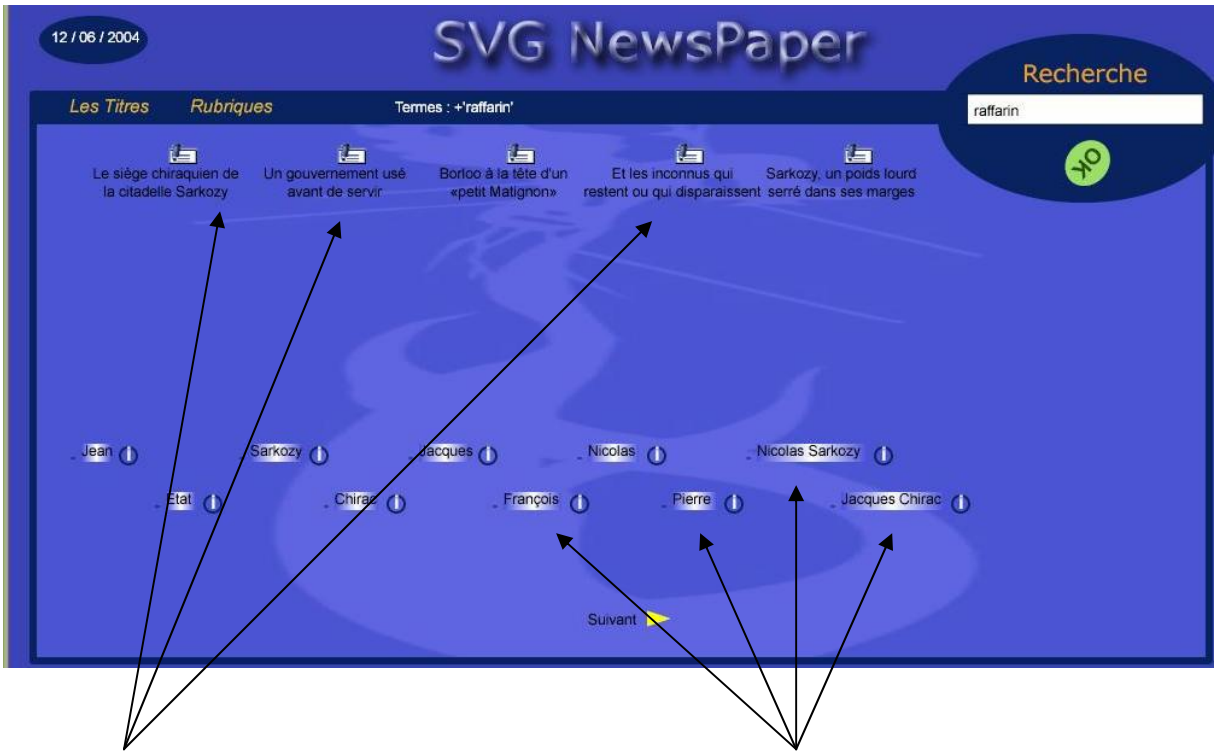


Pour cela, il lui suffit de taper sa requête dans la zone de recherche :

Exemple de recherche

Un clic sur

permet d'afficher les résultats de la recherche. Ceux-ci se décomposent en deux parties. D'abord, les articles concernés par les mots de la recherche, et ensuite, les principaux termes contenus dans ces articles.

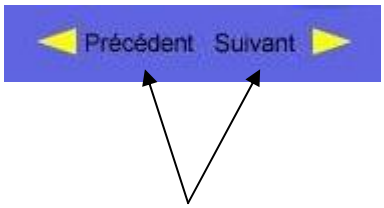


Articles concernés par le mot « raffarin »

Termes contenus dans les articles

Il peut également effectuer une recherche à partir des « termes du jour ». Pour cela, il lui suffit de cliquer sur celui qui l'intéresse.

Les résultats d'une recherche ne peuvent contenir que cinq articles. Si il y a plus de cinq articles concernés par la requête, l'utilisateur peut les consulter grâce au bouton « Suivant ». Il pourra revenir à la page d'origine en cliquant sur « Précédent ».

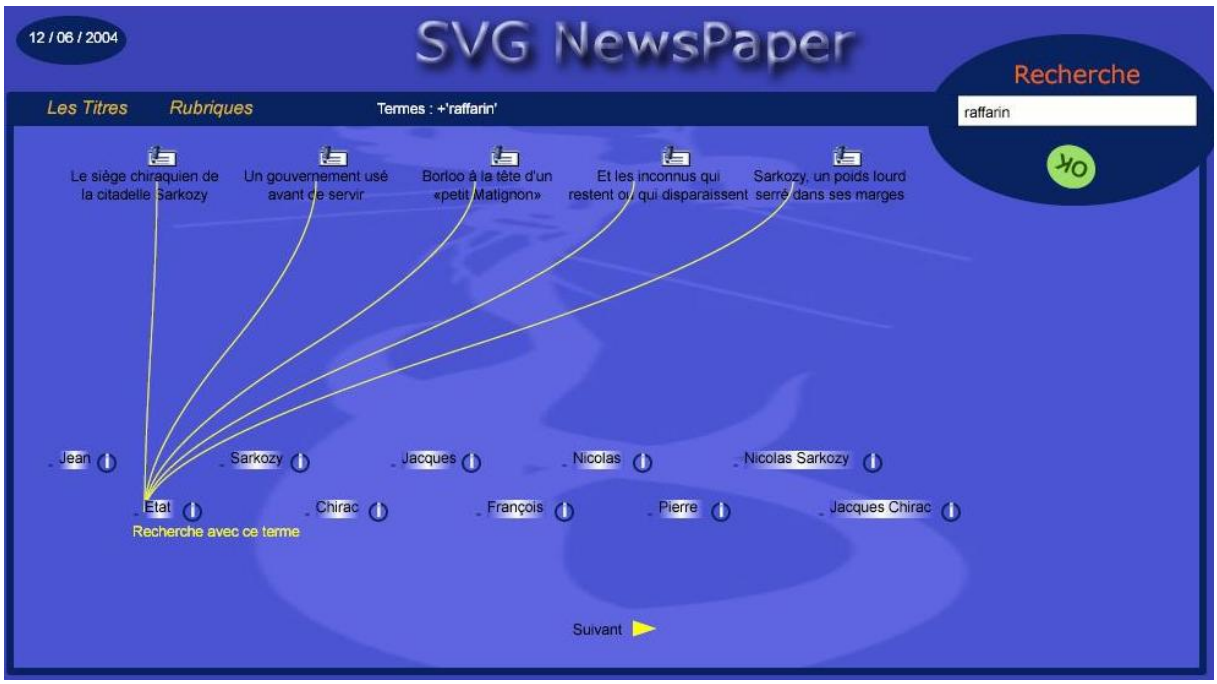


Boutons permettant de passer d'une page à l'autre

Il existe deux manières pour visualiser les liens entre les articles et les termes. Tout d'abord, en plaçant le curseur sur un article. Les courbes qui apparaissent permettent de savoir quels termes sont contenus dans l'article :

Liens entre un article et les termes qu'il contient

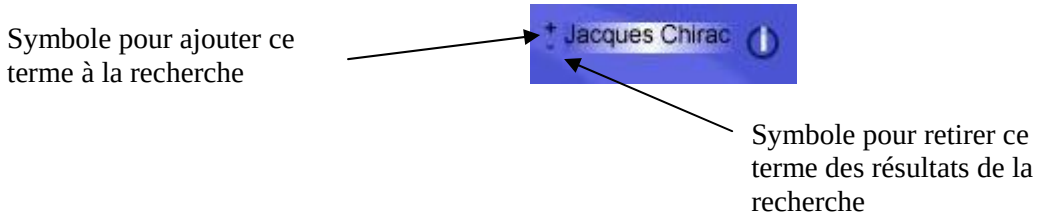
Puis, il est également possible d'obtenir des liens en plaçant le curseur sur un terme. Les courbes permettent alors de voir les articles qui contiennent ce terme :



Liens entre un terme et les articles qui le contiennent

Nouvelle recherche à partir des résultats

Si dans les résultats obtenus, certains termes n'intéressent pas l'utilisateur, il peut les retirer. Pour cela, il doit cliquer sur le symbole « - » qui se trouve à la gauche de celui-ci. Lorsque le curseur de la souris se trouve sur ce symbole, la phrase « Retirer de la recherche » apparaît. :



Une nouvelle recherche est alors lancée mais le résultat obtenu ne contient pas le terme retiré. Une partie de l'interface est prévu pour rappeler à l'utilisateur les termes de sa requête (précédé par un « + ») ainsi que ceux qu'il a retiré (précédé par un « - ») :

L'utilisateur peut aussi ajouter un terme à ceux qu'il a entré. Pour cela, il doit cliquer sur le symbole « + ».

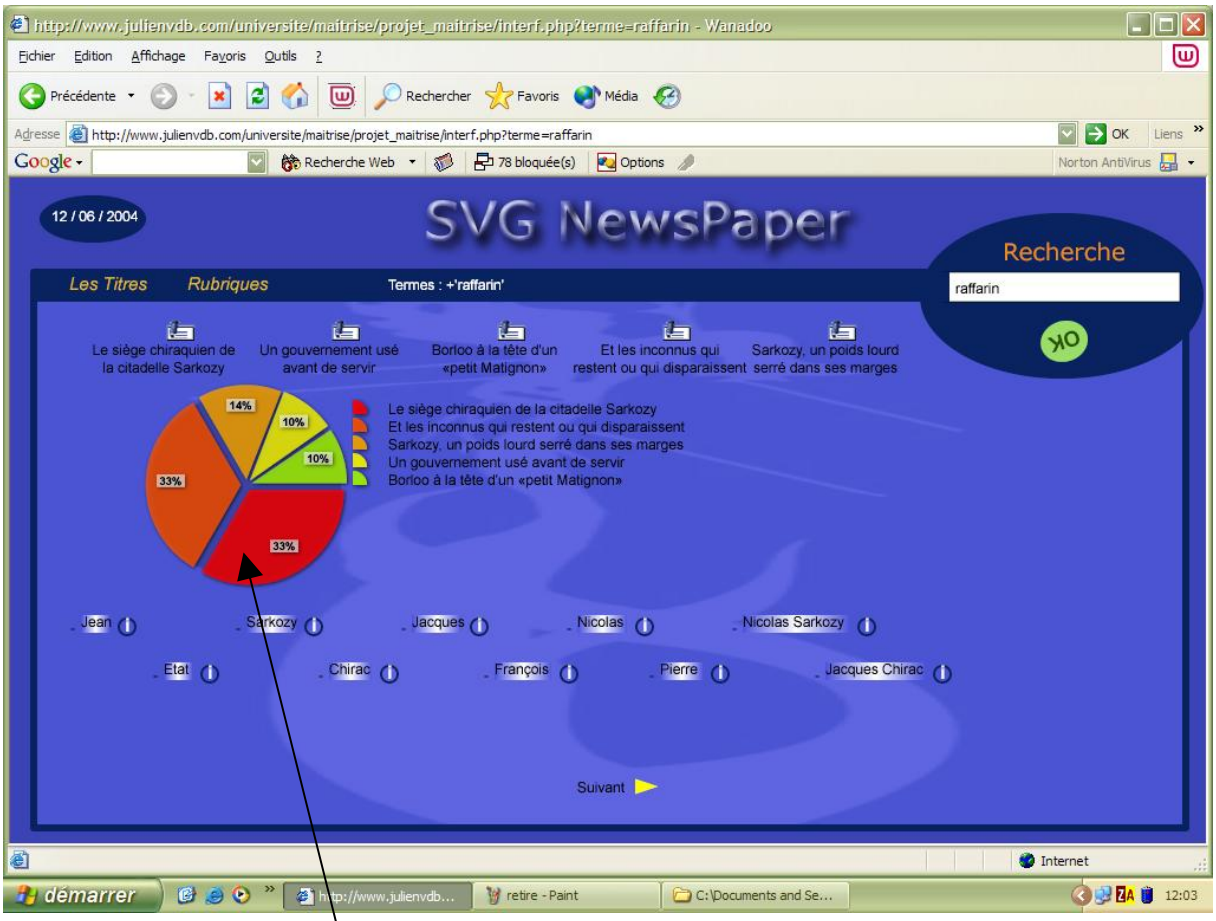
L'utilisateur a également la possibilité de lancer une nouvelle recherche à partir des termes qu'il a obtenu. Il doit donc cliquer sur le terme qu'il l'intéresse.

Une nouvelle page s'affiche dans laquelle le mot sélectionné se trouve dans la zone de recherche :

Le mot sélectionné se retrouve ici

Informations sur les termes

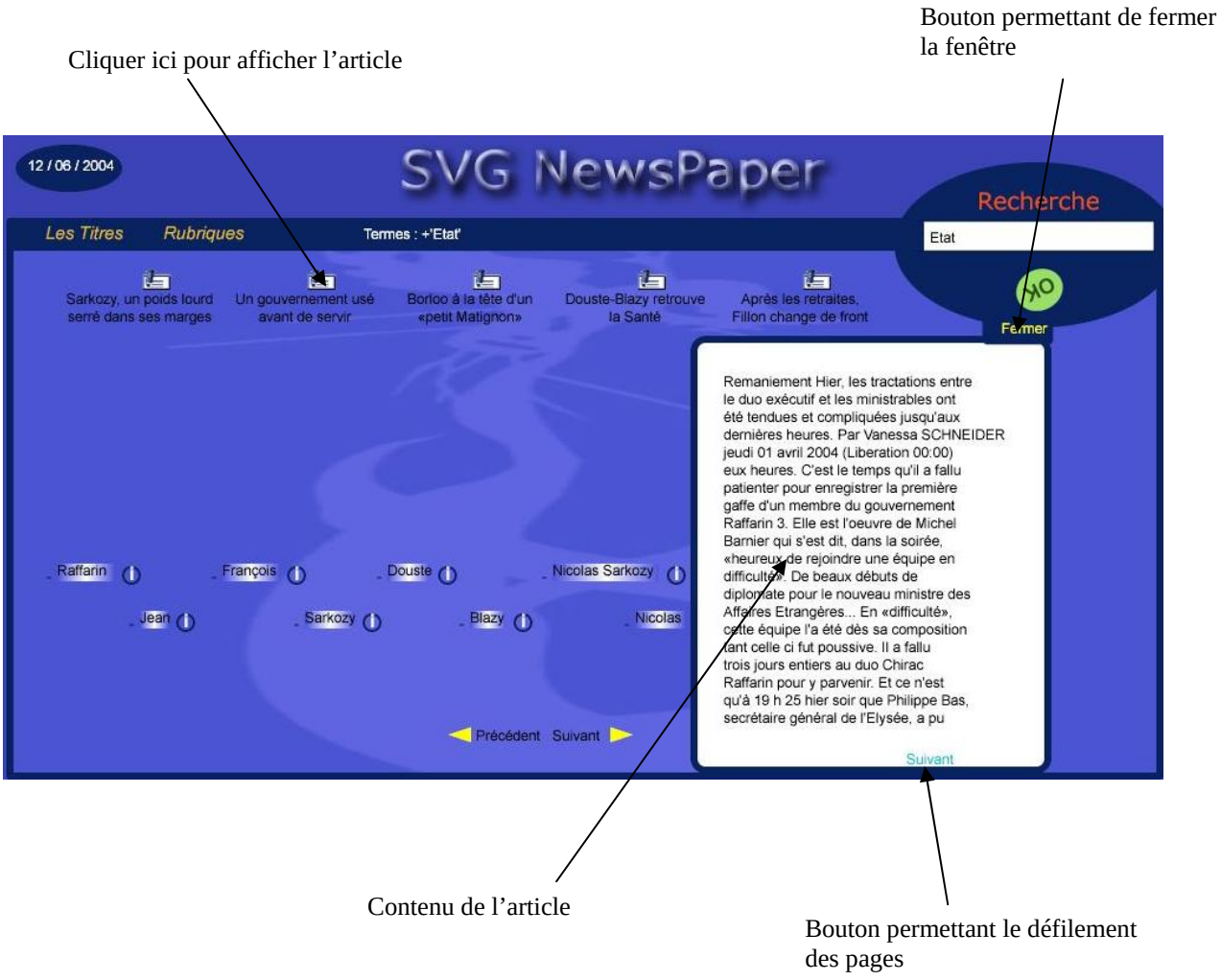
Pour obtenir des informations sur la présence et la fréquence des termes dans les articles, l'utilisateur doit cliquer sur l'icône qui se situe à la droite du terme. Un camembert avec des pourcentages apparaît. Plus le pourcentage est élevé, plus l'article parle du terme :



Pourcentage représentant l'utilisation du mot « Etat » dans l'article « Le siège chiraquien de la citadelle Sarkozy »

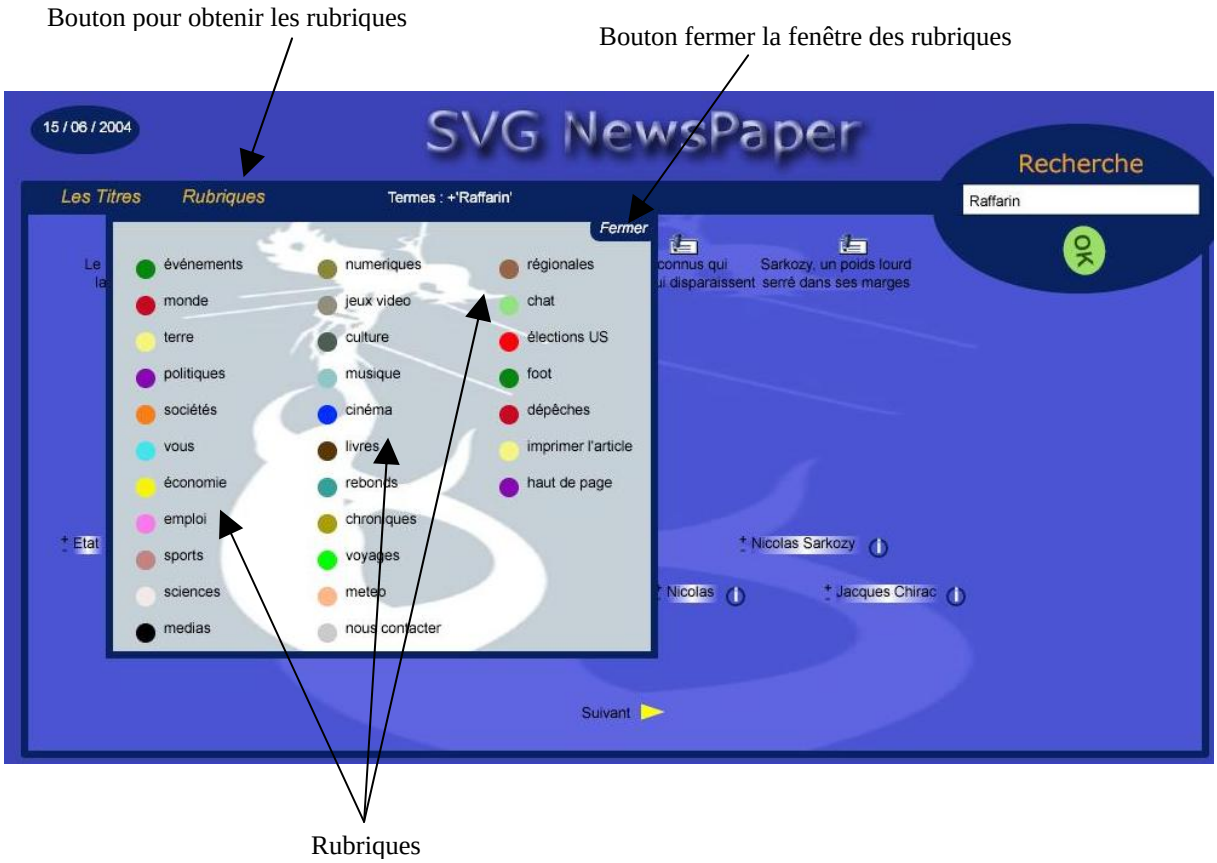
Affichage des articles

Pour afficher un article, il faut cliquer sur l'image située au-dessus du titre de l'article. Ensuite, pour faire défiler les pages de l'article, il faut utiliser les boutons « Précédent » et « Suivant ». Le bouton « Fermer » sert à faire disparaître la fenêtre de l'article.



Affichage des rubriques

Pour obtenir les différentes rubriques, l'utilisateur doit cliquer sur le bouton prévu à cet effet. Une fenêtre contenant les rubriques apparaît. Pour la refermer, il lui suffit de cliquer sur le bouton « Fermer ».



Retour à la page d'accueil

Pour revenir à la page d'accueil où se trouvent les termes du jour, il faut cliquer sur le bouton « Les Titres ».

Bouton pour revenir à la page « Les Termes du Jour »